One Framework

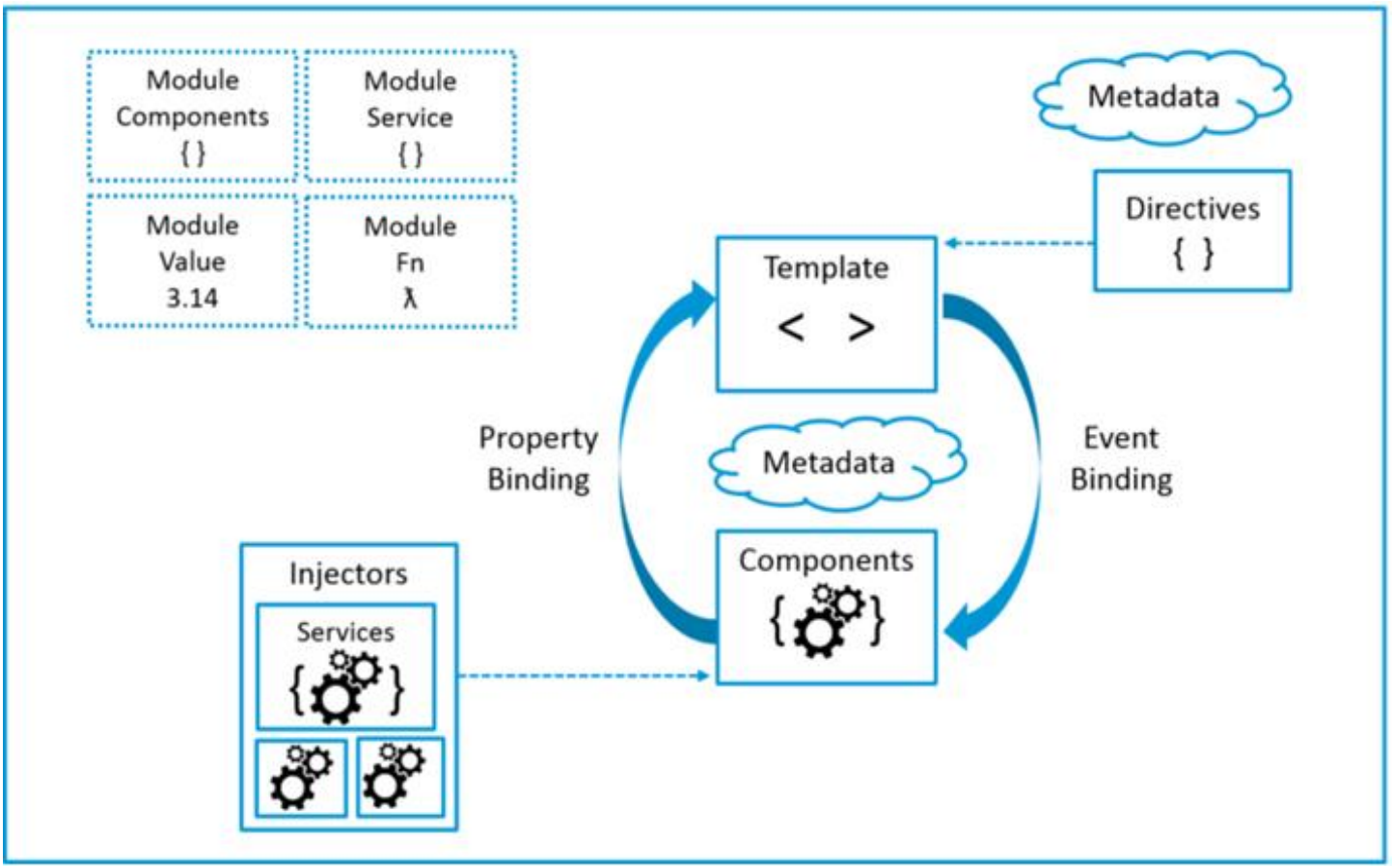
Mobile & desktop

Edition 2020

Sudhakar Sharma

Angular Architecture

Angular Architecture:-



Angular Framework component

Framework-component	Description
- Module	- A model is a
- Service	- A services is a predefined business logic with a set of factories It uses single ton mechanism
- Component	- A component comparison of UI, style and logic. That can handle specific functionality in user interface.
- Directive	- It makes more declarative it convert static DOM element into dynamic.

- Pipe	- A pipe is a predefined function used to filter, sort and format the data. Angular allows to create custom pipes.
- Selector	- It is responsible for handling a component in UI, which Include injecting the component, redirection, etc.
- Template	- It is a predefine component that can be rolled out across various pages.
- Material	- It is provide predefine component plugin which is you can used in your angular application.
- Animation	- It provides a library to handle 2D & 3D css animation in angular application.
- Routing	- It is technique used to configure user and SEO friendly navigation in application
- DI (Depending Injection)	- It is software design pattern used for testability in application.

Environmental Setup for Angular

1. Install Node JS - - - - - - - www.nodejs.org
2. Install Typescript - - - - - - - npm install –g typescript
3. Install Angular CLI - - - - - - npm install @Angular/cli@latest

Test Environment

- node –v [For node js]
- npm –v [For package manager]
- tsc [For Typescript]
- ng version [For angular version]

Upgrade from older version to Angular 12

1. Uninstall existing angular cli
   - npm uninstall –g @angular-cli
2. Clear cache

- npm cache clear
- npm cache verify

3. Install latest version
   - npm install –g @angular/cli@latest
4. Check the version of angular
   - ng version

Create a New Angular Project:-

1. Create a new Folder on your drive
   - “c:\angular8project”
2. Open folder in your VS code
3. Open terminal “ Ctrl+` ” (Ctrl + back tic)
4. Type the command
   - ng new shopping
5. Add routing model to project? : n
6. Which css format you want? : css

Angular project Infrastructure:-

Component	Descripton
- e2e	- It comprises “end to end” configuration for angular application, which include development, quality and performance.
- node_modules	- It is a repository of all library install in your project. [It belong to npm]
- Src	- It comprises of project resources like components, services, pipes assets [image, pdf] etc.
- .editorconfig	- It sets configuration for editor (VS Code) to used angular project. It contains charset, spacing style, indends etc.
- .gitignore	- It contains configuration to collaborate the current application with git repository.

	- It contains the metadata of current angular application. It include version, style, scripts and Various environment.
- Browerlist	- Sets configuration for browser supported for your application.
- karma.conf.js	- It contain configuration for testing environment.
- package.json	- It provides the list of libraries used in application and allow to restore library. [npm install –to restore]
- package-lock.json	- it contain meta data of package installed.
- tsconfig.json	- Typescript configuration for compatibility.
- tsconfig.app.json	- TS configuration for current application.
- tsconfig.spec.json	- It contain test configuration for ts.
- tslint.json	- It contain the rules reference for Typescript used in application

Resources in “app” folder: -

Resource Name	Description
- app.module.ts	- It is startup configure file which comprises of all configuration setting that are required to start angular application.
- Assets	- It comprises of non-dynamic file such as images and other static documents.
- environment [src]	- It contains files that are required for starting application on various environments, ie. Development, testing and production.
- index.html	- it is the startup document, angular application allows start with index page.
- main.ts	- It setup environment for development and testing angular application.
- polyfills.ts	- It defines configuration for browsers used in application

- style.css	- It contains global CSS ie. Style accessible to any component
- test.ts	- It contains configuration for testing framework.

Setting in "app.module.ts":-

Sr.no	Property	Description
1	declaration [ ]	- It is a property used to Register the component of angular application.
2	import [ ]	- It is the property used to Register module used in angular application
3	providers [ ]	- It is a property used to Register all services used in angular application.
4	bootstrap [ ]	- Its specifies the components to startwith.

Syntax:-

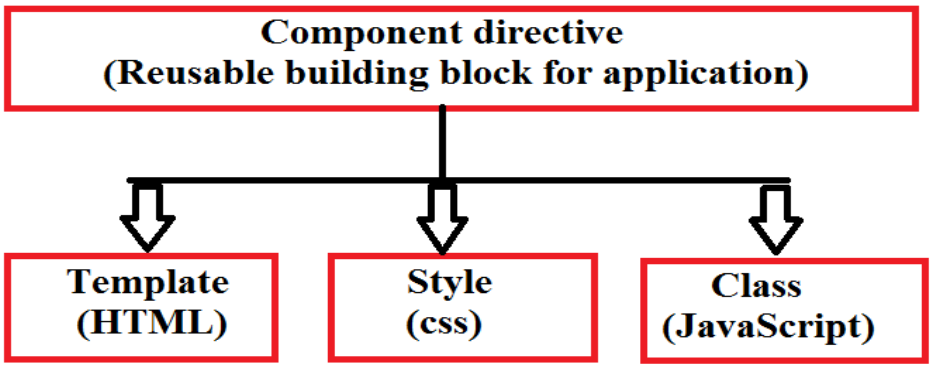
```
@NgModule({
  declarations: [
    Your_component,
  ],
  imports: [
    Your_module,
  ],
  providers: [],
  bootstrap: [Component_to_start]
})
```

To Start Angular application: -

1. Open terminal Ctrl+`
2. Change your project directory
3. Type the following command
   - ng serve or
   - ng s or
   - npm start
4. This will start a live server where angular application is hosted
5. Open any browser and request the following " http://localhost:4200"

Angular Component

- Component are the building blocks for angular application.
- A component in angular is technically a class that is responsible for handling a template.
- A template comparison of UI, Style, and logic.



- A component is configure by using the directive @ component ( ).
- The component directive is derived from "component" interface.
- The component interface is configure in "@angular/core" library.
- To configure any component it required the following properties in metadata.

Component	Description
- Selector	- It specifies the direction name used to access the component in any page.
- Template	- It specifies the markup for template to render when component is access in page.
- Template URL	- It specifies the page that contains markup to be rendered (.html)
- Style	- It specifies a set of style attribute used for markup in the component.
- StyleUrl	- It specifies the stylesheat used for component (.css)
- Animation	- It defines a set of animations, which have css 2D and 3D animations used for html element in the component.

Syntax: -

```
@Component({
  selector: 'app-name'  template: '<markup>',
  templateUrl: 'page.html',
  styles: ['select{attribute : value}'],
  styleUrls: ['stylesheet.css'],
  animations:[ 'transition, transform ..']
})
```

- Here we can Adding a component by using 3 technique those are following
    a) Using Inline technique
    b) Using Code behind Technique
    c) Using CLI command
- Example 1

EX-1 Home

*Adding a component using Inline Technique

1. Goto “app” folder in “src”
2. Add a new file - - - - - - - - - - - - “ home.Component.ts ”

```
import { Component } from '@angular/core';
@Component ({
    selector: 'app-home',
    template: `
                <h2>{{msg}}</h2>
                <p>This is our Home Component</p>
                `,
    styles : ['h2{color:red}']
})

export class HomeComponent {
    public msg = 'Welcome to Angular 12';
}
```

3. Goto “ app.module.ts ”

```
import { HomeComponent } from './home.component';

@NgModule({
  declarations: [
    HomeComponent
```

```
  ],
bootstrap: [HomeComponent]
})
```

4. Goto " index.html "

```
<body>
  <app-home></app-home>
</body>
```

5. Start your project
   - ng s –o
6. Output Display

Note- Every time go to index.html write in body which file you want open for these exam selector is "app-home" so in index page in body section write <app-home> </app-home>

- Example 2

EX-2 Register

1. Add a new folder into "app" folder
   - "register"
2. Add following files into register folder
   - register.component.ts
   - register.component.html
   - register.component.css
3. register.Component.ts [Code]

```
import { Component } from '@angular/core';

  @Component({
    selector: 'app-register',
    templateUrl: './register.component.html',
    styleUrls: ['./register.component.css']
  })
  export class RegisterComponent  {
    public title ='register user';
  }
```

4. register.Component.html

```
<div>
    <h2>{{title}}</h2>
    <label>Email</label>
</div>
<div>
    <input type="email">
    <button>Register</button>
</div>
```

5. Register.Component.css

```
h2{
    background-color:darkcyan;
    color:white;
    text-align: center;
}
input[type="email"]{
    border:solid 2px darkcyan;
}
button{
    background-color: darkcyan;
    color: white;
}
```

6. Output :

- Adding a new Component using CLI Command

1. Used the following command in terminal
   - ng generate component anyName
   - ng g c anyName
   - gng g c anyName –spec=false

Summary

- Angular Architecture
  - Angular framework component.
  - Angular project infrastructure.
  - Resource in app folder
- Angular Component
  - Using Inline technique
  - Using Code behind Technique
  - Using CLI command

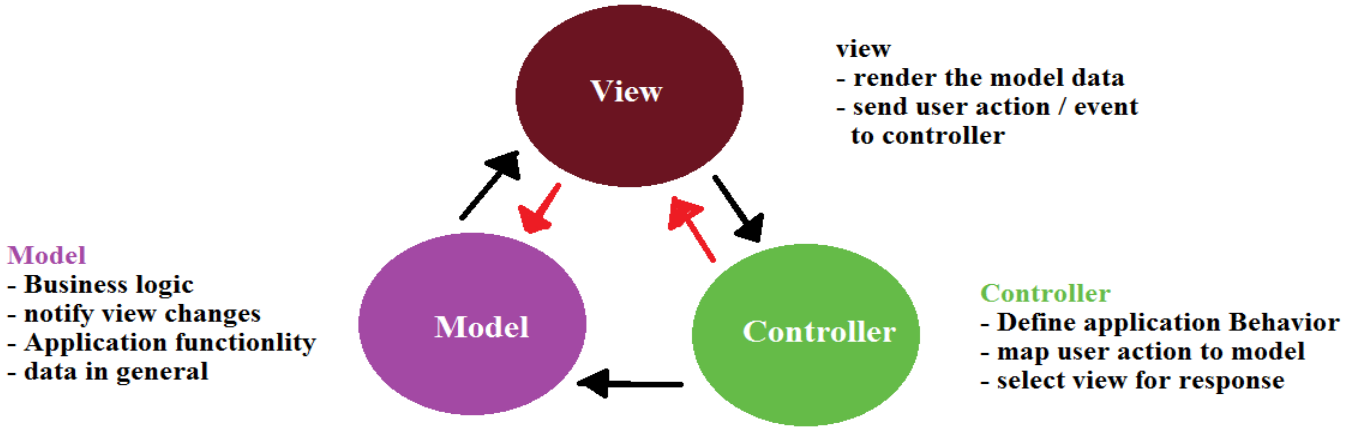
Data Binding

Data Binding:-

- Data binding is a technique that allow angular application to bind its model data to the view.
- Angular uses the framework like MVC, and MVVM to handle data binding.
- The Framework like MVC and MVVM introduced feature like-
  - Code reusability
  - Code separation
  - Maintainability
  - Extensibility
  - Testability

MVC Framework:-

- MVC is an software Architecture pattern introduce early 1970's by "Trygve".
- MVC introduced code reusability and code separation.
- MVC separate the application into 3 major components.
  - Model
  - View
  - Controller



1. **Model: -**

- It represent the data we are working with as well as the business.
- It uses POJO [Plain-old JavaScript object].
- It contains the data as well as the schema [Data Structure]

2. **View: -**

- It description the application UI.
- It dynamically renders HTML.
- It presents the Nodel data.
- It comprises of views and partial view.

3. **Controller: -**

- It is the core MVC component
- It control the overall application flow.
- It handle communication between model and view.
- It comprises of application specific logic client side it's a collection of function and events.

MVVM Framework:-

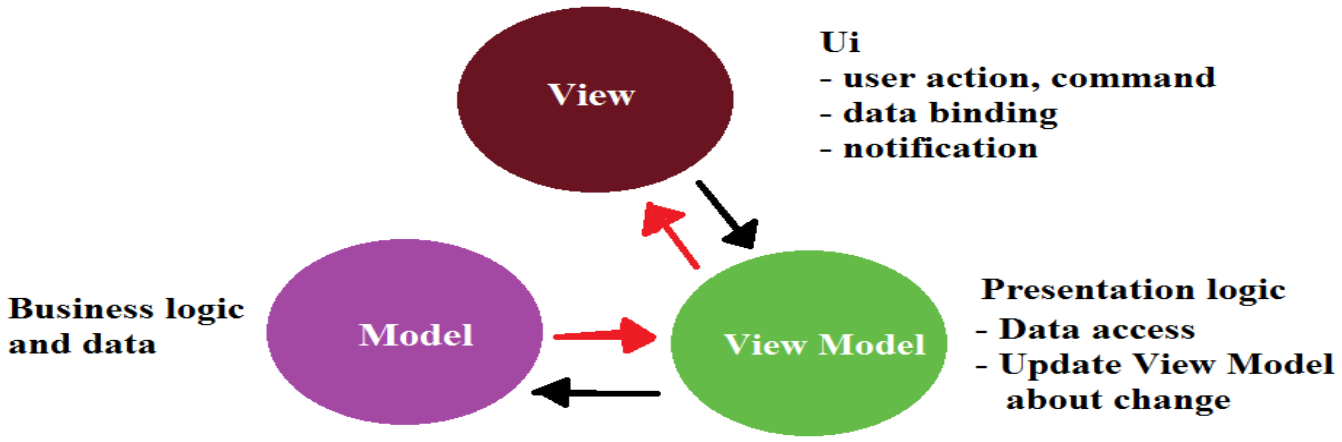


Fig: - MVVM framework

- In MVVM framework a view model is responsible for handling the overall communication between view and model.
- It can directly update the model data to view and vice-versa.
- View model is consider as "Single source of truth" in software Architecture.

Data Binding Technique:-

- The client side framework support data binding by using following technique.
  - ➢ One way binding.
  - ➢ Two way binding

1. **One way Binding: -**

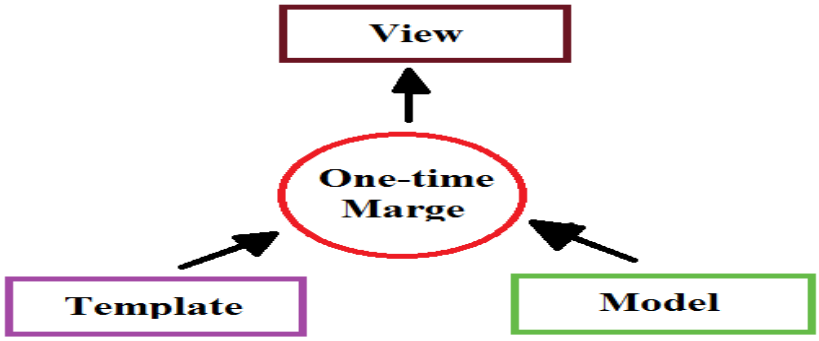


Fig:- One way Binding

- One way binding is a read only technique that's reads the model data bind to view.
- It is one time rendering.
- It will not update the view change to model data

2. **Two way Binding: -**

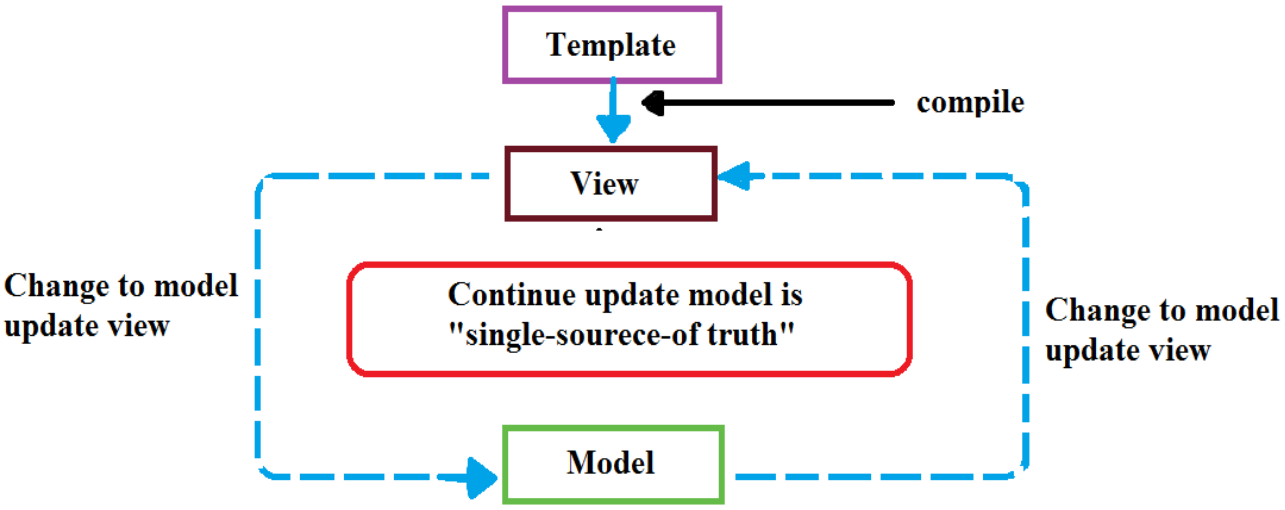


Fig:- Two way Binding

- Two way binding represent read and write operation handled on data.
- It is a technique that bind the model data to view and updates the view changes to model.

Angular Data Binding Technique:-

- Angular Support data binding by using 3 different technique
    1. Interpolation
    2. Property Binding
    3. Attribute Binding

1. Interpolation

- It is data binding technique used by angular to bind the model data to view by using data binding expression "{{ }}".
- It dynamically evaluate a value and binds to view
- It support only one way binding technique that is it cannot update the view changes into model.

Ex.3:- Databinding

1. Add n new Component
    - ng g c databinding –spec=false
2. Goto "databinding.Component.ts"

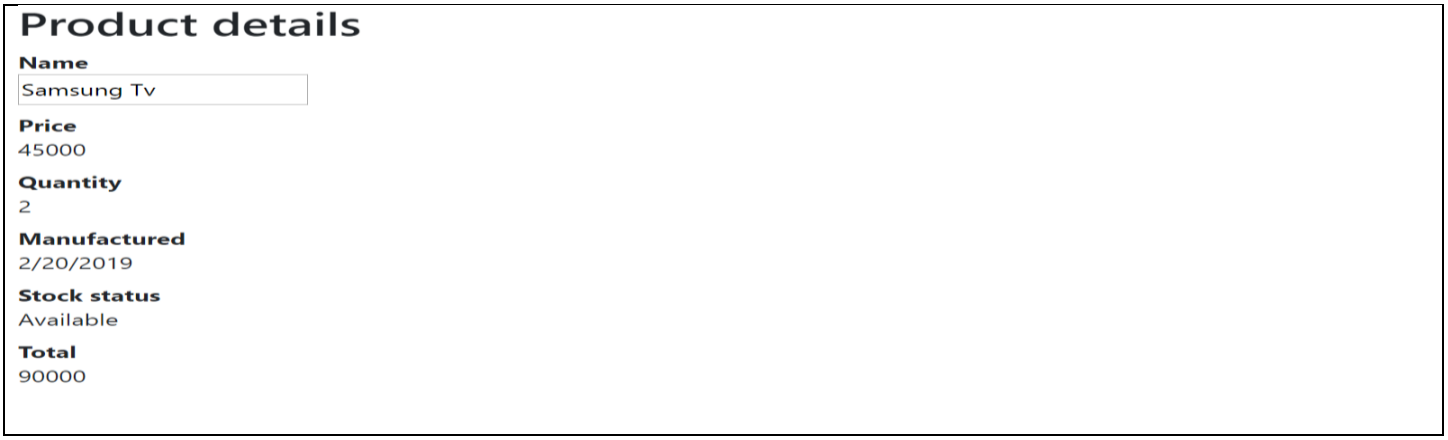
```
export class DatabindingComponent  {
    public product = {
      Name : 'Samsung Tv',
      Price : 45000,
      Qty : 2,
      mfd : new Date('2019/02/20'),
      InStock : true
    }
}
```

3. Goto "databinding.Component.html"

```
<h2>Product details</h2>
<dl>
<dt>Name</dt>
<dd><input type="text" value="{{product.Name}}"></dd>
<dt>Price</dt>
<dd>{{product.Price}}</dd>
<dt>Quantity</dt>
<dd>{{product.Qty}}</dd>
<dt>Manufactured</dt>
```

```
<dd>{{product.mfd.toLocaleDateString()}}</dd>
<dt>Stock status</dt>
<dd>{{(product.InStock==true) ? "Available" : "out of stock"}}</dd>
<dt>Total</dt>
<dd>{{product.Qty*product.Price}}</dd>
</dl>
```

4. Output



2. Binding to Property

- The model data can be binded to any DOM element dynamically by referring to its property and closed in "[ ]" square blasé

  Syntax: -

```
public uname = 'John';
<input type="text" [val]="uname">
```

- Data binding to any property will allow the view changes to be updated into model however this requires additionally event binding.

3. Binding to Attribute

- HTML element are configure with properties and their tag are configure with attribute
- Every attribute doesn't have it relative property in html element.

  Example: -

```
var tab = document.getElementById('table');
tab.width="200";  //valid
tab.colspan="2";  // invalid-colspan is not a property
```

- The dynamic data can be binded to attribute by using the prefix "altr"

Syntax: -

```
public margeCount =2;
<th [colspan]="margeCount">    // invalid
<th [altr.colspan]="margeCount">  //valid
```

Two Way Data Binding

- Angular handles two way binding by using a combination of property uses and event binding.
- The two way binding uses a data binding reference configured by using "[( )]"
  - [ ] – Property Binding
  - ( ) –Event Binding.
- The property binding bind any value to change view and event binding identifies the changes in view.
- Two way binding requires single source of truth (model) which identifies the model changes and updates to view.
- Angular configures a model "ngModel", ehich is a directive that keep tracking the changes.
- The "ngModel" directive is define in "form module" class of "@angular/form" library.
- Syntax:-

```
<input type="text" [(ngModel)]="ModelValue">
```

- Example 4 : -

Ex.4: - Databinding2

1. Goto app.module.ts

```
import { FormsModule } from '@angular/forms';

imports: [
FormsModule
]
```

2. Goto "databinding2.Component.ts "

```
export class Databinding2Component  {
public Name;
public Price;
```

```
public Shippedto;
public IsInStock;
}
```

3. Goto “databinding2.Component.html ”

```
<div>
    <h2>Register product</h2>
    <dl>
        <dt>Name</dt>
        <dd><input type="text" [(ngModel)]="Name"></dd>
        <dt>Price</dt>
        <dd><input type="text"[(ngModel)]="Price"></dd>
        <dt>Is in Stock</dt>
        <dd>
             <input type="radio" name="Stock" value="Available"
[(ngModel)]="IsInStock"> Yes
            <input type="radio" name="Stock" value="Out of stock" [(ngMode
l)]="IsInStock"> No
        </dd>
        <dt>Shipped To</dt>
        <dd>
            <select [(ngModel)]="ShippedTo">
                <option>Delhi</option>
                <option>Hyd</option>
            </select>
        </dd>
    </dl>
</div>
<div>
    <h2>Product Details</h2>
    <dl>
        <dt>Name</dt>
        <dd>{{Name}}</dd>
        <dt>Price</dt>
        <dd>{{Price}}</dd>
        <dt>Stock status</dt>
        <dd>{{IsInStock}}</dd>
        <dt>Shipped to</dt>
        <dd>{{ShippedTo}}</dd>
    </dl>
</div>
```

4. Goto “databinding2.Component.css ”

```
dt{
    font-weight:bold;
}
```

```
div{
    width: 300px;
    border: 2px solid red;
    padding: 10px;
    margin: 10px;
    margin-left: 10px;
    border-radius: 10;
}
```

5. Output

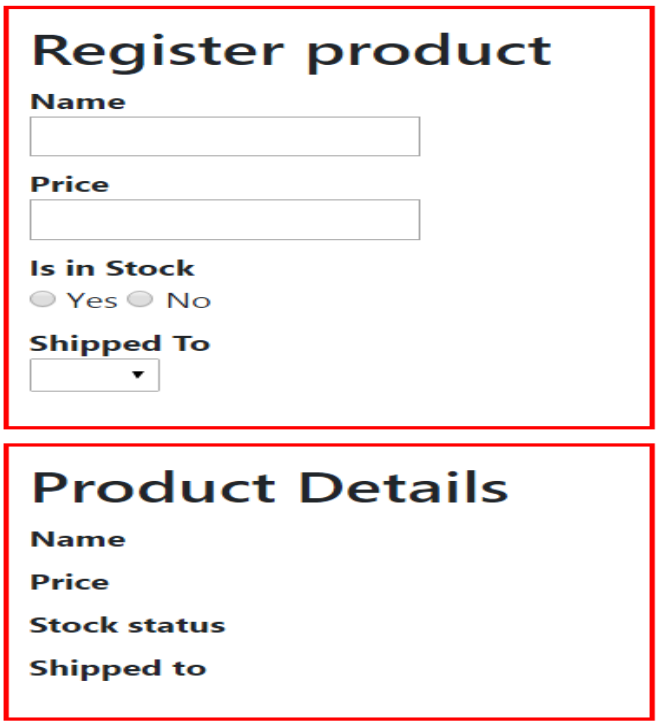


Fig.1 Before submit value

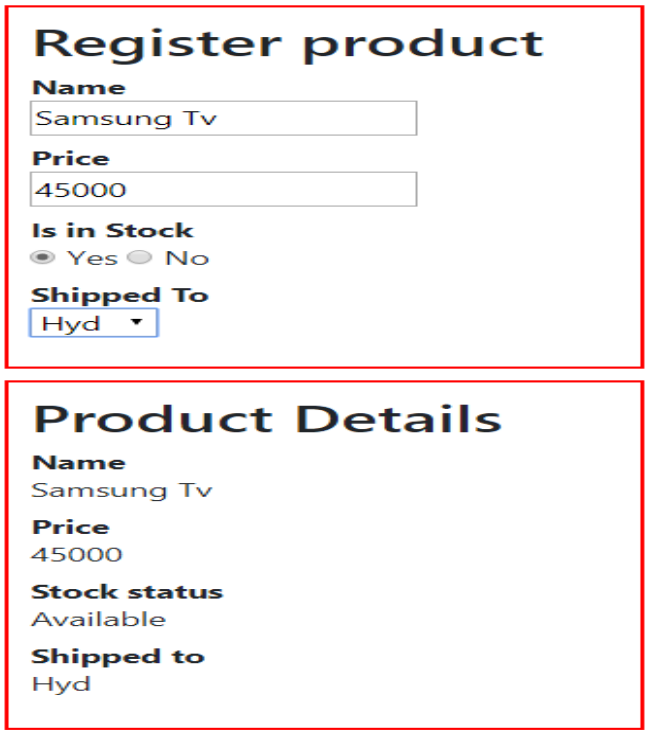


Fig.2 After submit value

Configure bootstrap for Angular Application

1. Install bootstrap in your project.
   - Shopping > npm install bootstrap
2. The bootstrap code library for css is in
   - node_module
     | _ bootstrap
       | _ dist
         | _ css
           | _ bootstrap.css

3. Goto “src/app/style.css” file and import @import “../node_module/bootstrap/dist/css/bootstrap.css”
4. You can apply bootstrap css classes directly to element in any component.html.
    - `<input type="text" class="form-group">`
    - `<button class="btn btn-primary"></button>`

Angular Directive

- A directive is a responsible for converting the static DOM element into dynamic.
- It is responsible for extending the markup and making html more declarative.
- Angular provides several pre-defines directives. Which are used to handle various interaction dynamically.
- The directive in angular are category into 3 types.
    1. Component
    2. Structural directive
    3. Attribute directive

1. Component :-

- The component directive are most common used directive for building an application.
- Component directive is Technology a template.
- It is used for rendering html dynamically
- Ex. Home.component.html

2. Structural directive: -

- The structural directive are responsible for changing the structure of view (UI)
- A structure directive can dynamically handle any html component, which include adding or removing element and conditionally adding the element.
- The Angular structural directive are
    - ngIf
    - ngFor
    - ngSwitch

- The directive are added to any html element by using an " * " (astric) and it comprises of conditional or iteration statement.

  Syntax: -

- Note:- the directive of angular are derived from same name module in angular library
- i.e. ngIf –module name "ngIf"
- ngIf - <ng-if> early version of Angular

A. ngIf directive: -

- It is an angular directive used to control the appearance of any element dynamically in the UI.
- It is responsible for adding or removing element DOM from DOM hierarchy.
- It is uses the Boolean "true" for adding and "false" for removing element.
- Syntax: -

```
<div *ngIf="boolean Value">
</div>
```

- Example 5 : -

Ex.5: - ifdemo

1. Add a new Component
   - ng g c ifdemo - -spec=false
2. Goto "ifdemo.Component.ts "

```
export class IfdemoComponent  {
    public isVisible = false;
    public btnText = 'Show';

    public product = {
      Name : 'samsung TV',
      Price : 3400.45,
      Photo : 'assets/tv1.jpg'
    };

  public DisplayClick() {
    this.isVisible = (this.isVisible == false) ? true : false;
    this.btnText = (this.btnText == 'Show') ? 'Hide' : 'Show';
  }
}
```

3. Goto "ifdemo.Component.html "

```
<div class="container">
    <h2>Product details</h2>
    <dl>
        <dt>Name</dt>
        <dd>{{product.Name}}</dd>
        <dt>Price</dt>
        <dd>{{product.Price}}</dd>
        <dt>
            <button (click)="DisplayClick()" class="btn btn-
primary">preview->{{btnText}}</button>
        </dt>
        <dd>
            <div *ngIf="isVisible">
                <img [src]="product.Photo" width="200" height="200" class=
"img-thumbnail">
            </div>
        </dd>
    </dl>
</div>
```

4. Output

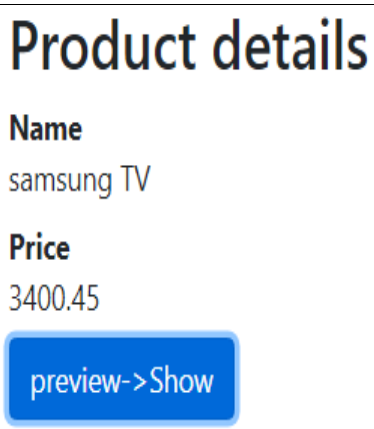


Fig.1 Before press button

Fig.2 After press button

B. ngFor directive: -

- it is an angular directive used to repeat any DOM element based on the specified iteration.
- The iteration for the directive are defined by using “of” operator
- Syntax:-

```
<div *ngFor="let item of collection">
    {{item}}
</div>
```

- Example 6 : -

- Example 7 : -
- Example 8 : -

Ex.6: - Fordemo

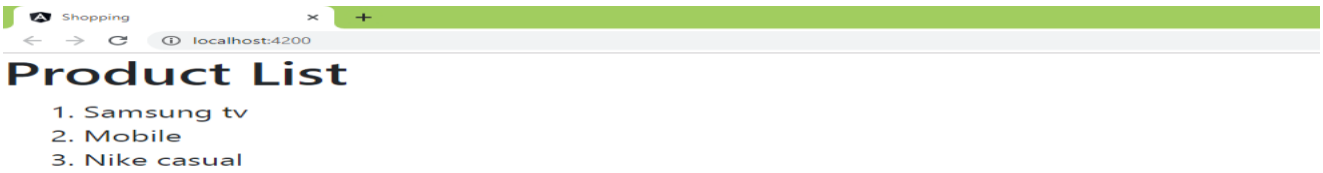
1. Add a new Component
   - ng g c fordemo - -spec=false
2. Goto “fordemo.Component.ts ”

```
export class FordemoComponent  {
  public product = ['Samsung tv','Mobile','Nike casual']
}
```

3. Goto “fordemo.Component.html ”

```
<div>
<h2>Product List</h2>
<ol>
<li *ngFor="let product of product">{{product}}</li>
</ol>
</div>
```

4. Output



Ex.7: - ngForNested

1. Add a new Component
   - ng g c ngfornested - -spec=false
2. Goto “ngfornested.Component.ts ”

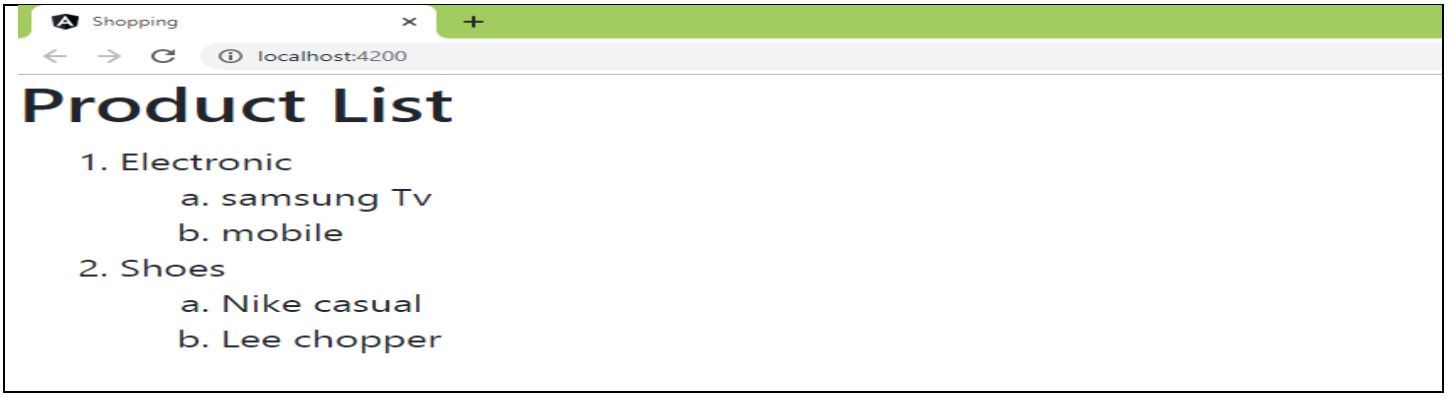
```
export class NgfornestedComponent  {
    public data = [
      {  category : 'Electronic',
        product : ['samsung Tv', 'mobile']
      },
```

```
        { category : 'Shoes',
          product : ['Nike casual','Lee chopper']
        }
      ];
  }
```

3. Goto “ngfornested.Component.html”

```
<div>
    <h2>Product List</h2>
    <ol>
        <li *ngFor="let item of data">
            {{item.category}}
            <ol type="a">
                <li *ngFor="let product of item.product">{{product}}

                </li>
            </ol>
        </li>
    </ol>
</div>
```

4. Output



Ex.8: - Catlog

1. Add a new Component
   - ng g c catlog - -spec=false
2. Goto “cattlog.component.ts ”

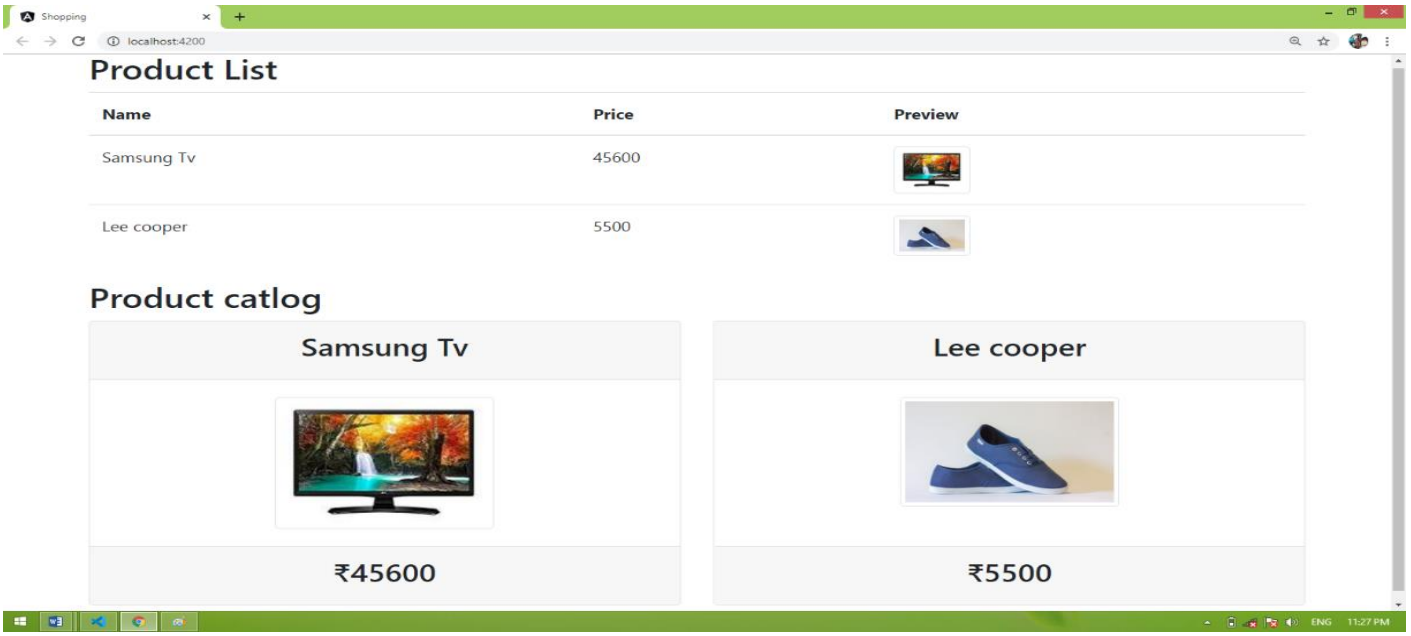
```
export class CatlogComponent {
public product = [
  {Id : 1, Name : 'Samsung Tv', Price : 45600, Photo : 'assets/tv1.jpg'},
```

```
  {Id : 2, Name : 'Lee cooper', Price : 5500, Photo : 'assets/shoe1.jpg'}
];
}
```

3. Goto “catlog.component.html”

```
<div class="container">
    <div class="form-group">
        <h2>Product List</h2>
        <table class="table table-hover">
            <thead>
                <th>Name</th>
                <th>Price</th>
                <th>Preview</th>
            </thead>
            <tbody>
                <tr *ngFor="let product of product">
                    <td>{{product.Name}}</td>
                    <td>{{product.Price}}</td>
                    <td >
                        <img [src]="product.Photo" width="70" height="70"
class="img-thumbnail">
                    </td>
                </tr>
            </tbody>
        </table>
    </div>
    <div class="form-group">
        <h2>Product catlog</h2>
        <div class="card-deck text-center" >
            <div class="card" *ngFor ="let product of product">
                <div class="card-header">
                    <h3>{{product.Name}}</h3>
                </div>
                <div class="card-body">
                    <img [src]="product.Photo" width="200" height="200"
class="img-thumbnail">
                </div>
                <div class="card-footer">
                    <h3> &#8377;{{product.Price}} </h3>
                </div>
            </div>
        </div>
    </div>
</div>
```

4. Output



*ngFor properties and Methods:-

Member	Type	Description
index	Number	- It returns iterator element index number starting with zero
first	Boolean	- It returns true if the element in iteration is the first item
last	Boolean	- It returns true if the element in iteration is the last item
even	Boolean	- It returns true if the item present at even occurrence
odd	Boolean	- It returns true if item present at odd occurrence
trackby	Function	- It uses a function that verify changes in the iteration and allow the itration only when changes occurance

- Example 9 : -Trackby
- Example 10 : - ngFor using index property
- Example 11 : - using the object reference to identify iteration counter
- Example 12 : - Apply effects based on even and odd occurrence

Ex.9: - trackby

1. Add a new Component
   - ng g c ifdemo - -spec=false
2. Goto “trackby.component.ts”

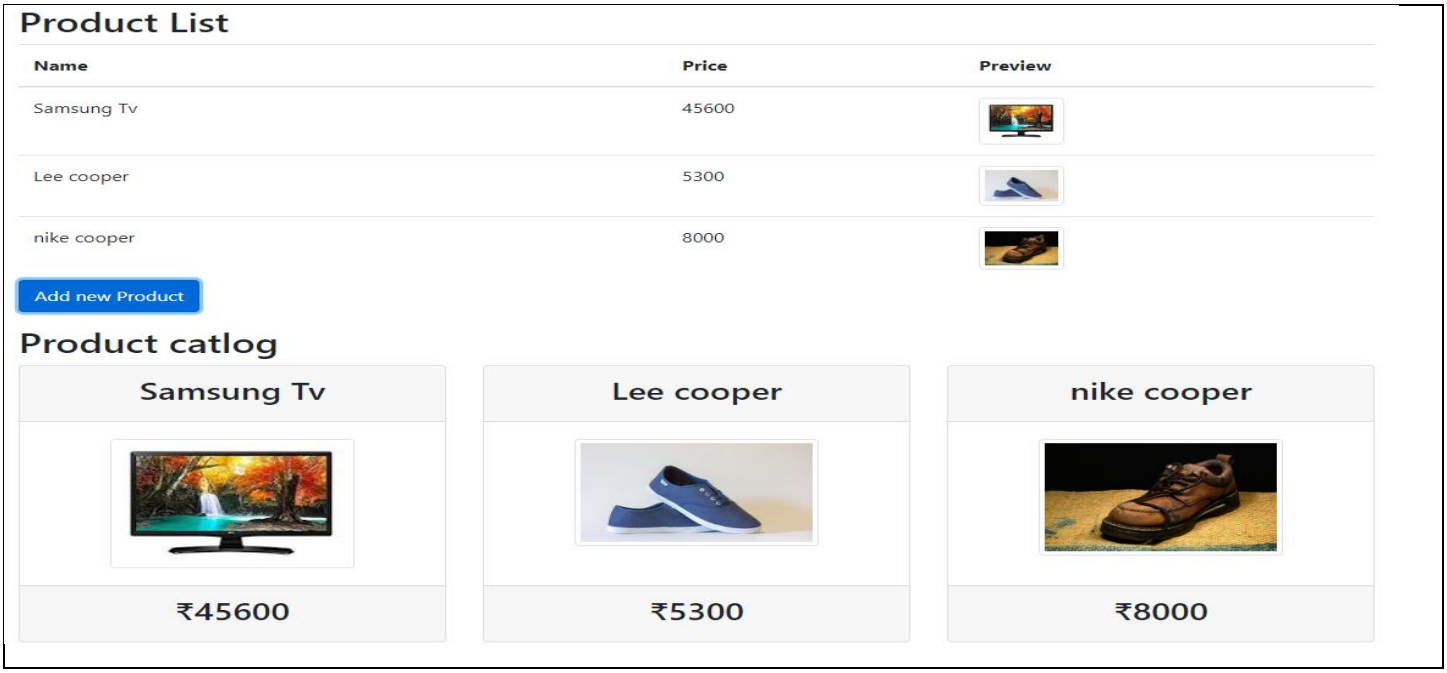
```
export class TrackbyComponent {
  public product = [
    { Id : 1, Name : 'Samsung Tv', Price : 45600, Photo : 'assets/tv1.jpg' },
    { Id : 2, Name : 'Lee cooper', Price : 5300, Photo : 'assets/shoe1.jpg' },
  ];
  public Addproduct(){
    this.product = [
    { Id : 1, Name : 'Samsung Tv', Price : 45600, Photo : 'assets/tv1.jpg' },
    { Id : 2, Name : 'Lee cooper', Price : 5300, Photo : 'assets/shoe1.jpg' },
    { Id : 3, Name : 'nike cooper',  Price : 8000 , Photo : 'assets/shoe2.jpeg'},
    ];
  }
    public TrackById(index, product) {
      return product.Id;
    }
}
```

3. Goto “trackby.component.html”

```
<div class="container">
<div class="form-group">
   <h2>Product List</h2>
   <table class="table table-hover">
     <thead>
       <th>Name</th>
       <th>Price</th>
       <th>Preview</th>
     </thead>
     <tbody>
       <tr *ngFor="let product of product ; trackBy:TrackById ">
          <td>{{product.Name}}</td>
          <td>{{product.Price}}</td>
```

```
            <td>
             <img [src]="product.Photo" width="70" height="70" class="img-thumbnail">
            </td>
            </tr>
            <div>
    <button (click)="Addproduct()" class="btn btn-primary"> Add new Product </button>
            </div>
       </tbody>
     </table>
</div>
<div class="form-group">
  <h2>Product catlog</h2>
   <div class="card-deck text-center" >
     <div class="card" *ngFor ="let product of product">
       <div class="card-header">
           <h3>{{product.Name}}</h3>
       </div>
       <div class="card-body">
           <img [src]="product.Photo" width="200" height="200" class="img-thumbnail">
        </div>
         <div class="card-footer">
           <h3> &#8377;{{product.Price}} </h3>
         </div>
        </div>
       </div>
     </div>
  </div>
```

4. Output



Ex.10: - iterationdemo

1. Add a new Component
   - ng g c iterationdemo - -spec=false
2. Goto “iterationdemo.component.ts ”

```
export class IterationdemoComponent {
  public product =[
    { Name : 'Samsung Tv', Price : 45400.50},
    { Name : 'Mobile', Price : 12500},
  ];
  public txtName;
  public txtPrice;
  public Newproduct = {
    Name: " ",
    Price: 0
  }
  public Addproduct() {
    alert('product Added');
    this.Newproduct = {
      Name  : this.txtName,
      Price : this.txtPrice
    };
    this.product.push(this.Newproduct);
    this.txtName = '',
    this.txtPrice = ''
  }
    public DeleteProduct(index) {
    var Status = confirm ('Are you sure, waqnt to Dlete?');
    if (Status == true){
      this.product.splice(index,1);
    }
  }
}
```

3. Goto “iterationdemo.component.html”

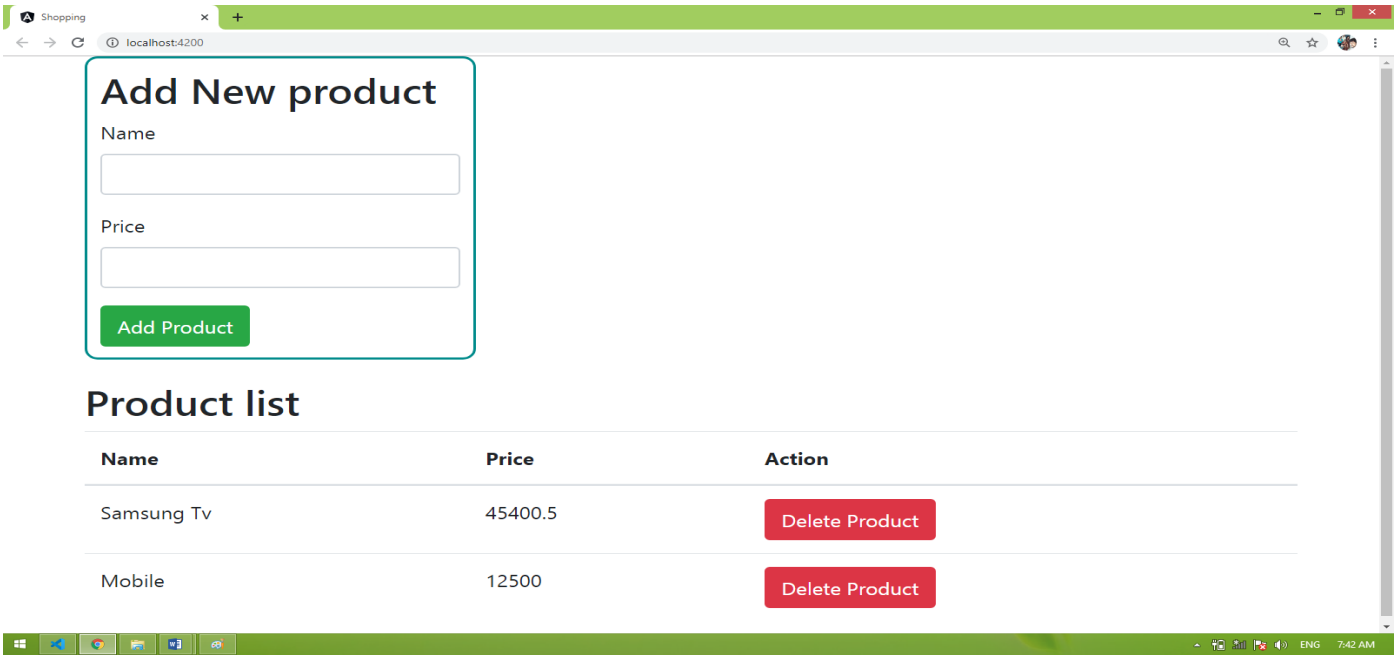
```
<div class="container">
    <div class="Addproduct">
        <h2>Add New product</h2>
        <div class="form-group">
            <label>Name</label>
            <div>
                <input [(ngModel)]="txtName" type="text" class="form-
control">
            </div>
```

```
        </div>
        <div class="form-group" >
            <label>Price</label>
            <div>
                <input [(ngModel)]="txtPrice" type="text" class="form-
control">
            </div>
        </div>
        <div >
            <button (click) ="Addproduct()" class="btn btn-
success"> Add Product </button>
        </div>
    </div>
    <div>
        <h2>Product list</h2>
        <table class="table table-hover">
            <thead>
                <th>Name</th>
                <th>Price</th>
                <th>Action</th>
            </thead>
            <tbody>
                <tr *ngFor="let product of product ; let i=index">
                    <td>{{product.Name}}</td>
                    <td>{{product.Price}}</td>
                    <td>
                        <button class="btn btn-
danger" (click)="DeleteProduct()"> Delete Product</button>
                    </td>
                </tr>
            </tbody>
        </table>
    </div>
</div>
```

4. Goto “iterationdemo.component.css”

```
.Addproduct{
    width:300px;
    border:2px solid darkcyan;
    border-radius:10px;
    padding:10px;
    margin-bottom:20px;
}
```

5. Output



Ex.11:- cars

1. Add a new Component
    a. ng g c cars - -spec=false
2. Goto “cars.component.ts”

```
export class CarsComponent {
  public cars = [
    { Name : 'Range Rower',Photo:'assets/car1.jpg', Like : 0, Dislike: 0 },
    { Name : 'Ferrari', Photo:'assets/car2.jpg', Like : 0, Dislike: 0 }
  ];
  public DislikeClick (car) {
      car.Dislike++;
  }
  public LikeClick (car) {
      car.Like++;
  }
}
```

3. Goto “cars.component.html”

```
<div class="containetr">
    <div class="card-deck">
        <div class="card" *ngFor="let car of cars">
            <div class="card-header">
                <h3>{{car.Name}}</h3>
```

```
            </div>
          <div class="card-body text-center">
               <img [src]="car.Photo" width="200" height="200" class="img-
thumbnail">
          </div>
          <div class="card-footer text-center">
              <button (click)="LikeClick(car)" class="btn  btn-success">
             [{{car.Like}}] Like
              </button>
              <button (click)="DislikeClick(car)" class="btn  btn-danger">
              [{{car.Dislike}}] Dislike
              </button>
          </div>
      </div>
  </div>
```

4. Output

Ex.12: - iterationdemo2:-

1. Add a new Component
    a. ng g c iterationdemo2 - -spec=false
2. Goto “iterationdemo2.component.ts”

```
export class Iterationdemo2Component  {
  public product =[
    { Name:'Samsung Tv', Price : 45400.50},
    { Name:'Mobile', Price : 12500},
  ];
  public txtName;
  public txtPrice;
  public Newproduct={
```

```
    Name:"",
    Price: 0
  }
  public Addproduct() {
    alert('product Added');
    this.Newproduct={
      Name  :this.txtName,
      Price : this.txtPrice
    };
    this.product.push(this.Newproduct);
    this.txtName='',
    this.txtPrice=''
  }
    public DeleteProduct(index) {
    var Status=confirm('Are you sure, waqnt to Dlete?');
    if(Status==true){
      this.product.splice(index,1);
    }
  }
}
```

3. Goto “iterationdemo2.component.html”

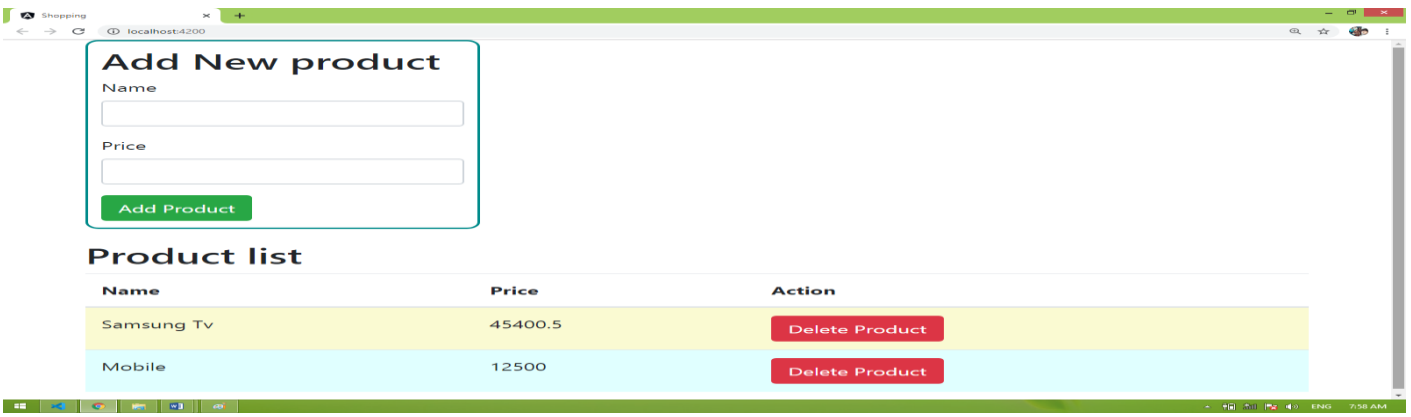
```
<div class="container">
    <div class="Addproduct">
        <h2>Add New product</h2>
        <div class="form-group">
            <label>Name</label>
            <div>
                <input [(ngModel)]="txtName" type="text" class="form-control">
            </div>
        </div>
        <div class="form-group" >
            <label>Price</label>
            <div>
                <input [(ngModel)]="txtPrice" type="text" class="form-control">
            </div>
        </div>
        <div>
            <button (click)="Addproduct()" class="btn btn-
success"> Add Product </button>
        </div>
    </div>
    <div>
        <h2>Product list</h2>
        <table class="table table-hover">
            <thead>
                <th>Name</th>
```

```
                <th>Price</th>
                <th>Action</th>
            </thead>
            <tbody>
                <tr [class.odd]="odd" [class.even]="even" *ngFor="let product of
product ; let i=index; let odd=odd; let even=even">
                    <td>{{product.Name}}</td>
                    <td>{{product.Price}}</td>
                    <td>
                        <button class="btn btn-
danger" (click)="DeleteProduct()"> Delete Product</button>
                    </td>
                </tr>
            </tbody>
        </table>
    </div>
</div>
```

4. Goto “iterationdemo2.component.css”

```
.Addproduct{
    width:300px;
    border:2px solid darkcyan;
    border-radius:10px;
    padding:10px;
    margin-bottom:20px;
}
.odd{
    background-color:lightcyan;
}
.even{
    background-color:lightgoldenrodyellow;
}
```

5. Output



Condition in iteration:-

- Angular supports the structural directive to handle various condition within an iteration.
- How ever angular will not allow multiple template binding in one element i.e. you cannot used multiple directive one element.
- You can handle the logic by using an angular container <ng-container>
- ngContainer is a directive that allow to handle a set of element without defining a markup.
- Syntax:-

```
<ng-container *ngIf="condition">
</ng-container>
```

- Example 13 : -

Ex.13:- condition

1. Add a new Component
    a. ng g c condition - -spec=false
2. Goto “condition.component.ts”

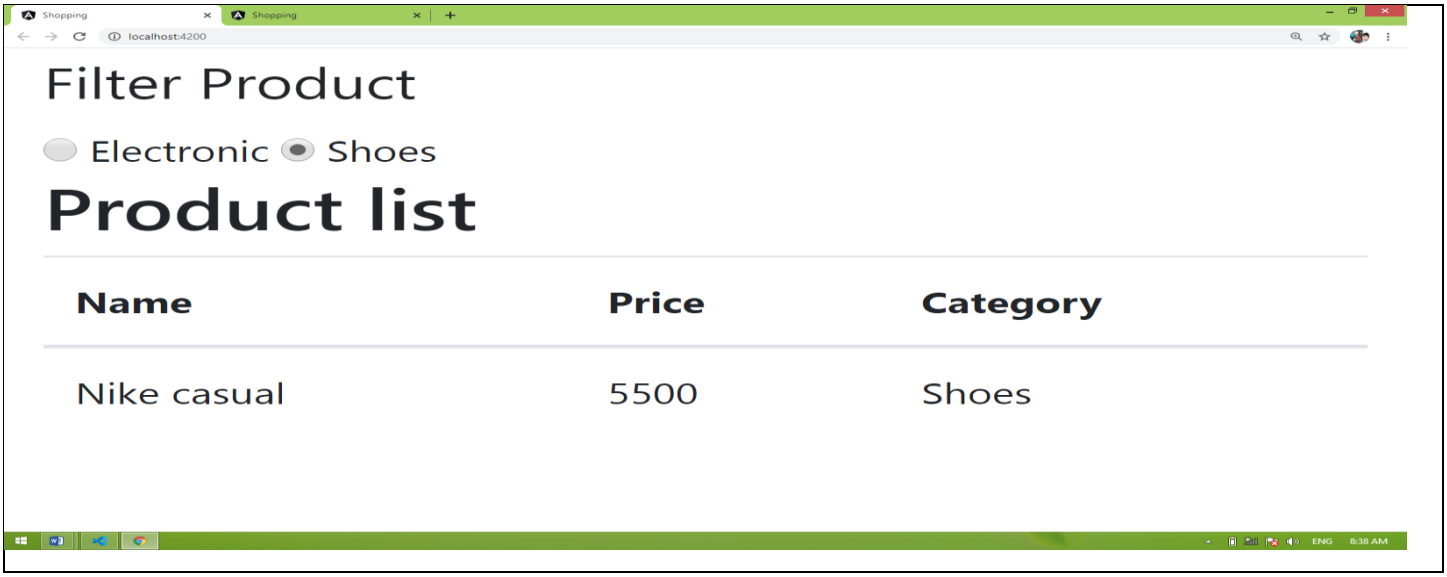
```
export class ConditionComponent  {
    public product =[
      {Name: 'Nike casual', Price: 5500, Category: 'Shoes'},
      {Name: 'Samsung Tv', Price: 54200, Category: 'Electronic'},
    ];
    public filter = 'Electronic'
}
```

3. Goto “condition.component.html”

```
<div class="container">
    <fieldset>
        <legend>Filter Product</legend>
        <input type="radio" [(ngModel)]="filter" name="filter" value="Electronic"
> Electronic
        <input type="radio" [(ngModel)]="filter" name="filter" value="Shoes">
Shoes
    </fieldset>
    <h2>Product list</h2>
    <table class="table table-hover">
        <thead>
            <th>Name</th>
            <th>Price</th>
            <th>Category</th>
```

```
            </thead>
            <tbody>
                <tr *ngFor="let product of product">
                    <ng-container *ngIf="product.Category==filter">
                        <td>{{product.Name}}</td>
                        <td>{{product.Price}}</td>
                        <td>{{product.Category}}</td>
                    </ng-container>
                </tr>
            </tbody>
        </table>
    </div>
```

4. Output



C. ngSwitch directive: -

- It is an angular structural directive used to handle multiple condition in the UI and display only the content that is required for the situation.
- Switch is used to interrupt the execution flow
- ngSwitch is a selector switch that execution only the block that matched the condition.
- The blocks that switch have to execute are define by using “ngSwitchCase”
- The default block have to execute when condition is not matching is defined by using “ngSwichdefault”
- Syntax: -

```
            <main-container [ngSwitch]="condition">
                <child-container *ngSwitchCase=" ">
                </child-container>
```

```
                <child-container *ngSwitchDefult>
            </main-container>
```

- Example 14 : - a)
- Example 14 : - b) Sequentially Accessing views.

Ex.14 a):- Switchdemo

1. Add a new Component switchdemo
    a. ng g c - -spec=false
2. Goto “switchdemo.component.ts”

```
export class SwitchdemoComponent  {
  public product = {
    Name:'Samsung Tv',
    Price:45000,
    Description:'Some text about Samsung Tv...',
    mfd: new Date() ,
    Photo:'assets/tv1.jpg'
  };
  public selectedView = 'info';

  public ChooseView(obj){
    this.selectedView=obj.target.value;
  }
}
```

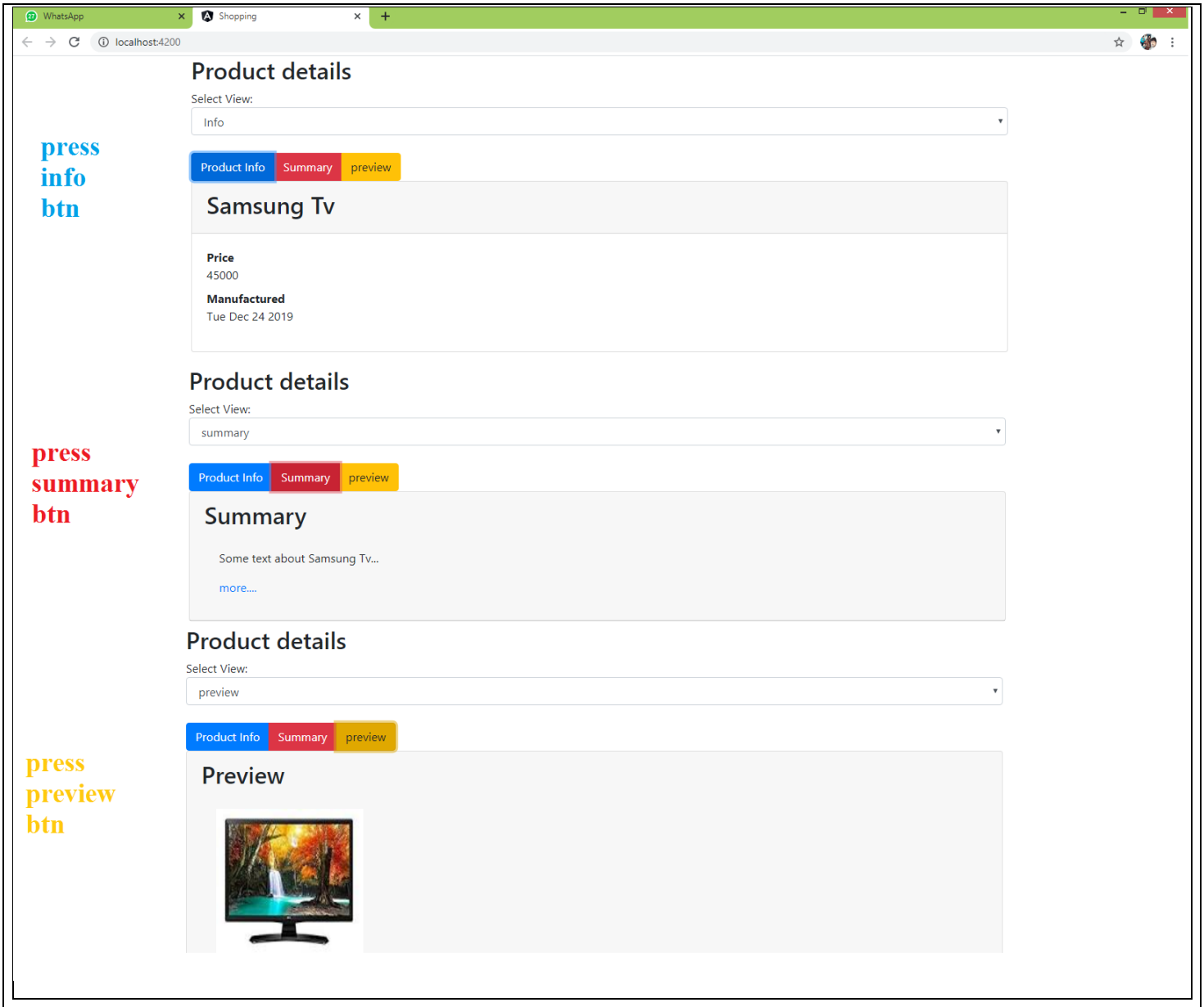
3. Goto “switchdemo.component.html”

```
<div class="container">
    <h2>Product details</h2>
    <div>
        Select View:
        <select [(ngModel)]="selectedView" class="form-control">
            <option value="info">Info</option>
            <option value="summary">summary</option>
            <option value="preview">preview</option>
        </select>
    </div>
    <br>
    <div class="btn-toolbar">
        <div class="btn-group">
            <button (click)="ChooseView($event)" value="info" class="btn btn-
primary">Product Info</button>
```

```
            <button (click)="ChooseView($event)" value="summary" class="btn btn-
danger">Summary</button>
            <button (click)="ChooseView($event)" value="preview" class="btn btn-
warning">preview</button>
        </div>
    </div>
    <div [ngSwitch]="selectedView">
        <div *ngSwitchCase="'info'"  class="card">
            <div class="card-header">
                <h2>{{product.Name}}</h2>
            </div>
            <div class="card-body">
                <dl>
                    <dt>Price</dt>
                    <dd>{{product.Price}}</dd>
                    <dt>Manufactured</dt>
                    <dd>{{product.mfd.toDateString()}}</dd>
                </dl>
            </div>
        </div>
        <div *ngSwitchCase="'summary'" class="card">
            <div class="card-header">
                <h2>Summary</h2>
                <div class="card-body">
                    <p>{{product.Description}}</p>
                    <a href="#" class="card-link"> more....</a>
                </div>
            </div>
        </div>
        <div *ngSwitchCase="'preview'" class="card">
            <div class="card-header">
                <h2>Preview</h2>
                <div class="card-body">
                    <img [src]="product.Photo" width="200" height="200">
                </div>
            </div>
        </div>
    </div>
</div>
```

4. Output

   Note:- in above example press random 3 buttons (in 1 image three screenshot available)



Ex.14 b):-Sequentially Accessing View

Note:- this example is same as previous example only Add some point in .ts file and change Button toolbar in .html file . In the output information is change dynamically by pressing button next or previous.

1. Add following into "switchdemo.component.ts "

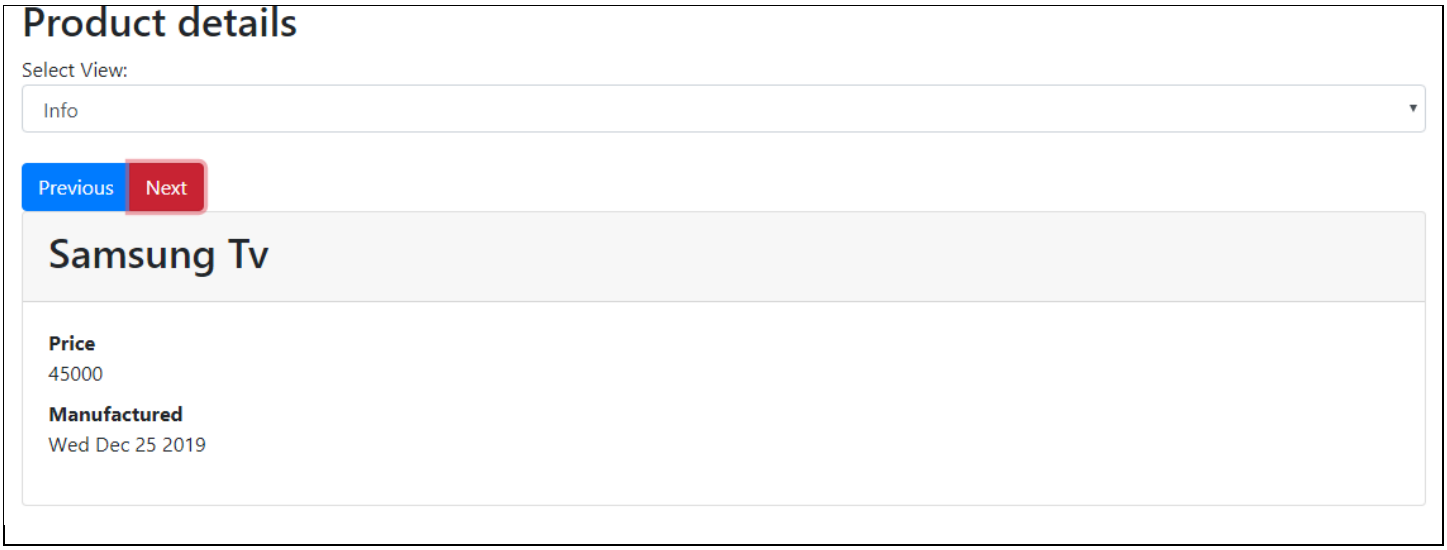
```
public selectedView = 'info';
  public view = ['info', 'summary', 'preview'];
```

```
  public index= 0;

  public Next(){
    this.index++;
    this.selectedView=this.view[this.index];
  }
  public Previous() {
    this.index--;
    this.selectedView=this.view[this.index]
  }
```

2. Change the button toolbar in “swichdemo.component.ts ”

```
<div class="btn-toolbar">
        <div class="btn-group">
            <button (click)="Previous()"  class="btn btn-
primary">Previous</button>
            <button (click)="Next()"  class="btn btn-danger">Next</button>
        </div>
    </div>
```

3. Output –press button previous and next .



- Class binding is a technology that allow to apply any css class to an Html element dynamically.
- The directive ngClass is used to bind any class to html element

3. Attribute Binding:-

1. Class Binding

- Class binding is a technique that allows to apply any css class to an html element dynamically.
- The directive ngClass is used to bind any class to HTML element.
- The class binding uses following reference technique.

a) String reference ‘ ‘

- To apply any specific css class.
- Syntax :-

```
<div [ngClass]="'className'">
</div>
```

b) Array reference [ ]

- To apply any multiple css class.
- Syntax :-

```
<div [ngClass]="['class1','class2'......]">
</div>
```

c) Object reference { }

- To turn on and off the css classes
- syntax :-

```
<div [ngClass]="[class1:true, class2:false]">
</div>
```

- Note: - Always used single quote (‘ ‘) for class name if it contains special chars.
- Syntax :-

```
<div [ngClass]="{'text-center':true}">
</div>
```

- Example 15: - classdemo

Ex.15:- classdemo

1. Add a new Component
    a. ng g c classdemo - -spec=false
2. Goto “classdemo.component.ts”

```
export class ClassdemoComponent  {
  public effects;
  public isBackEffectSelected;
  public isTexteffectSelected;
  public isBoederEffectSelected;
}
```

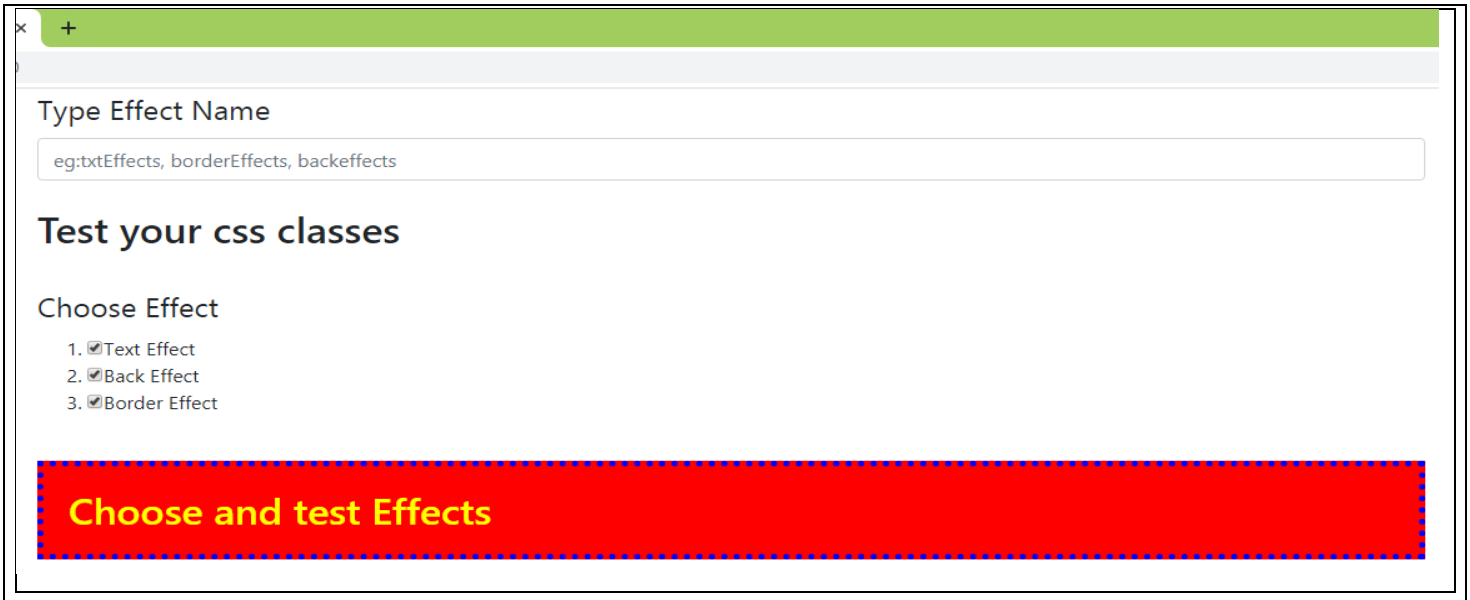
3. Goto “classdemo.component.html”

```
<div class="container">
    <fieldset>
        <legend>Type Effect Name</legend>
        <input [(ngModel)]="effects" placeholder="eg:txtEffects, borderEff
ects, backeffects" type="text" class="form-control">
    </fieldset>
    <br>
    <h2 [ngClass]="effects">Test your css classes</h2>
    <br>
    <legend>Choose Effect</legend>
    <ol>
        <li><input [(ngModel)]="isTextEffectSelected" type="checkbox">
Text Effect</li>
        <li><input [(ngModel)]="isBackEffectSelected" type="checkbox">
Back Effect</li>
        <li><input [(ngModel)]="isBorderEffectSelected" type="checkbox">
Border Effect</li>
    </ol>
    <br>
    <h2 [ngClass]="{
txtEffects:isTextEffectSelected,backEffects:isBackEffectSelected,'border-
Effects':isBorderEffectSelected}">Choose and test Effects</h2>
</div>
```

4. Goto “classdemo.component.css”

```
.txtEffects{
    color:yellow;
    padding: 20px;
}
.backEffects{
    background-color: red;
}
.border-Effects{
    border: 5px dotted blue;
}
```

5. Output



2. Style Binding

- It is a binding technique used by angular to bind inline style for any html element.
- It provide option that allow to change the style dynamically.
- The directive ngStyle is used to bind inline style. It required a style object with set of style attribute and values.
- Syntax:-

```
<div [ngStyle]="{attribute:value,......}">
</div>
```

- Example 16: - styledemo

Ex.16:- Styledemo

1. Add a new Component
    a. ng g c styledemo - -spec=false
2. Goto “styledemo.component.ts”

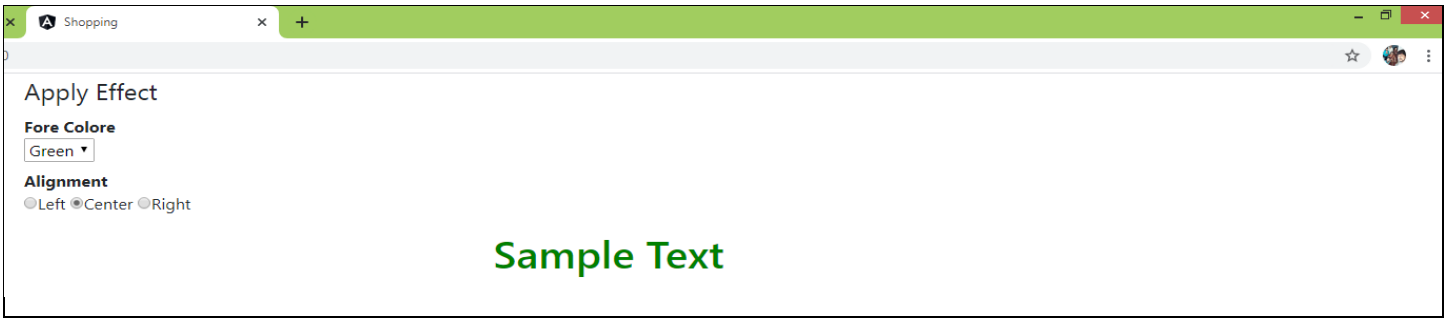
```
export class StyledemoComponent {
  public styleObject;
  public foreColor='black';
  public alignment='left';
  public ApplyEffects() {
    this.styleObject={
      'color':this.foreColor,
```

```
        'text-align':this.alignment
      };
      return this.styleObject;
    }
  }
```

3. Goto “styledemo.component.html”

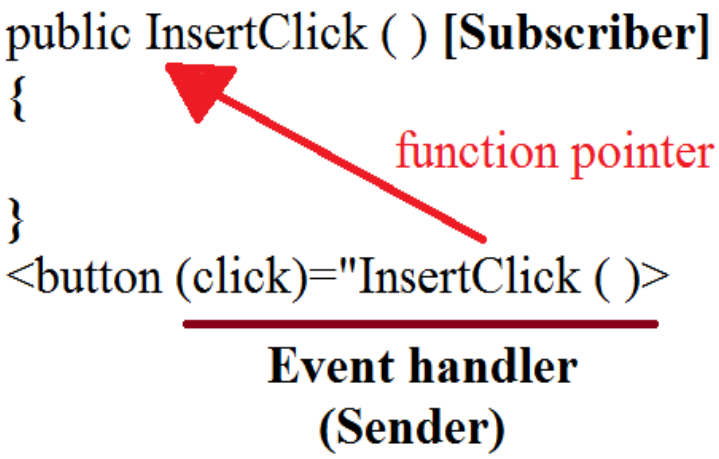
```
<div class="container">
    <fieldset>
        <legend>Apply Effect</legend>
        <dl>
            <dt>Fore Colore</dt>
            <dd>
                <select [(ngModel)]="foreColor">
                    <option value="green">Green</option>
                    <option value="black">Black</option>
                    <option value="red">Red</option>
                </select>
            </dd>
            <dt>Alignment</dt>
            <dd>
                <input [(ngModel)]="alignment" type="radio" name="align"
value="left">Left
                <input [(ngModel)]="alignment" type="radio" name="align"
value="center">Center
                <input [(ngModel)]="alignment" type="radio" name="align"
value="right">Right
            </dd>
        </dl>
    </fieldset>
    <h1 [ngStyle]="ApplyEffects()">Sample Text</h1>
</div>
```

4. Output – Select color from select box and Alignment from radio button it works.



Event Binding in Angular

- An object in oop comprises data in the form of properties and functionality in the form of methods.
- The method of an object or any component are executed (trigger) on specific event.(event trigger function).
- Event is a message send by sender to its subscriber in order to notify the change.
- It uses a event handler.



- Event handler follower the delegate mechanism, which is a function pointer to word the same signature function.
- In software programming event is design by following a design pattern called “observer”.
- Observer is a software communication pattern used to notify the changes and carry out the communication.
- Angular events are derived from “event emitter” base class.
- The event emitter emits its properties and methods by using “$event” which gives access to member of event.
- Syntax: -

```
public InsertClick(e) {
     e.clientX, e.clientY, e.ctrlkey
  }
 <button (click)="InsertClick($event)"></button>
```

Angular Event: -

- Angular support all browser event provided by the browser object which are common across all client side framework and library the events are categories into following type.
  1. Mouse event.
  2. Key event.
  3. Animation event.
  4. Timer event.
  5. Miscellaneous event.
  6. Custom events.

1. Change Event: -

- It trigger the action when value is changed in any HTML element. It is mostly suitable like number, range, dropdown, and text.
- Example 17 : - onlineshopping

Ex.17:- onlineshopping

1. Add a new Component
   - ng g c onlineshop - -spec=false
2. Goto "onlineshop.component.ts"

```
export class OnlineshoppingComponent {
  public categories = ['Select a Category', 'Electronics', 'Shoes'];
  public electronics = ['Select Electronic Product', 'Samsung TV', 'MI Mobile'];
  public shoes = ['Select Shoe Product', 'Nike Casuals', 'Lee Cooper Boot'];
  public selectedProducts = [];
  public selectedCategory;

  public productsData = [
    { Name: 'Samsung TV', Price: 45000.53,  Photo: 'assets/tv1.jpg'},
    { Name: 'MI Mobile', Price: 15000.53, Photo: 'assets/mobile1.jpeg' },
    { Name: 'Nike Casuals', Price: 6000.53,  Photo: 'assets/shoe1.jpg' },
    { Name: 'Lee Cooper Boot', Price: 3000.53,  Photo: 'assets/shoe2.jpeg'}
  ];
  public selectedProduct;
  public searchResults = [];
  public prodName= '';
  public prodPrice=0;
```

```
  public prodPhoto='';
  public cartItems = [];
  public cartItemsCount=0;
  public isCartVisible=false;

  public CategoryChanged() {
    switch(this.selectedCategory) {
      case 'Electronics':
        this.selectedProducts=this.electronics;
        break;
        case 'Shoes':
          this.selectedProducts = this.shoes;
          break;
    }
  }

  public ProductChanged() {
    this.searchResults = this.productsData.filter(x=>x.Name==this.selectedProduct
);
    this.prodName = this.searchResults[0].Name;
    this.prodPrice = this.searchResults[0].Price;
    this.prodPhoto = this.searchResults[0].Photo;
  }
  public AddToCart() {
    this.cartItems.push(this.searchResults[0]);
    this.cartItemsCount = this.cartItems.length;
  }
  public DeleteCartItem(index){
     let status = confirm('Are you sure, Want to Delete?');
     if(status==true) {
       this.cartItems.splice(index,1);
       this.cartItemsCount = this.cartItems.length;
     }
  }
  public ShowCart(){
    this.isCartVisible = (this.isCartVisible==false)?true:false;
  }
 }
```

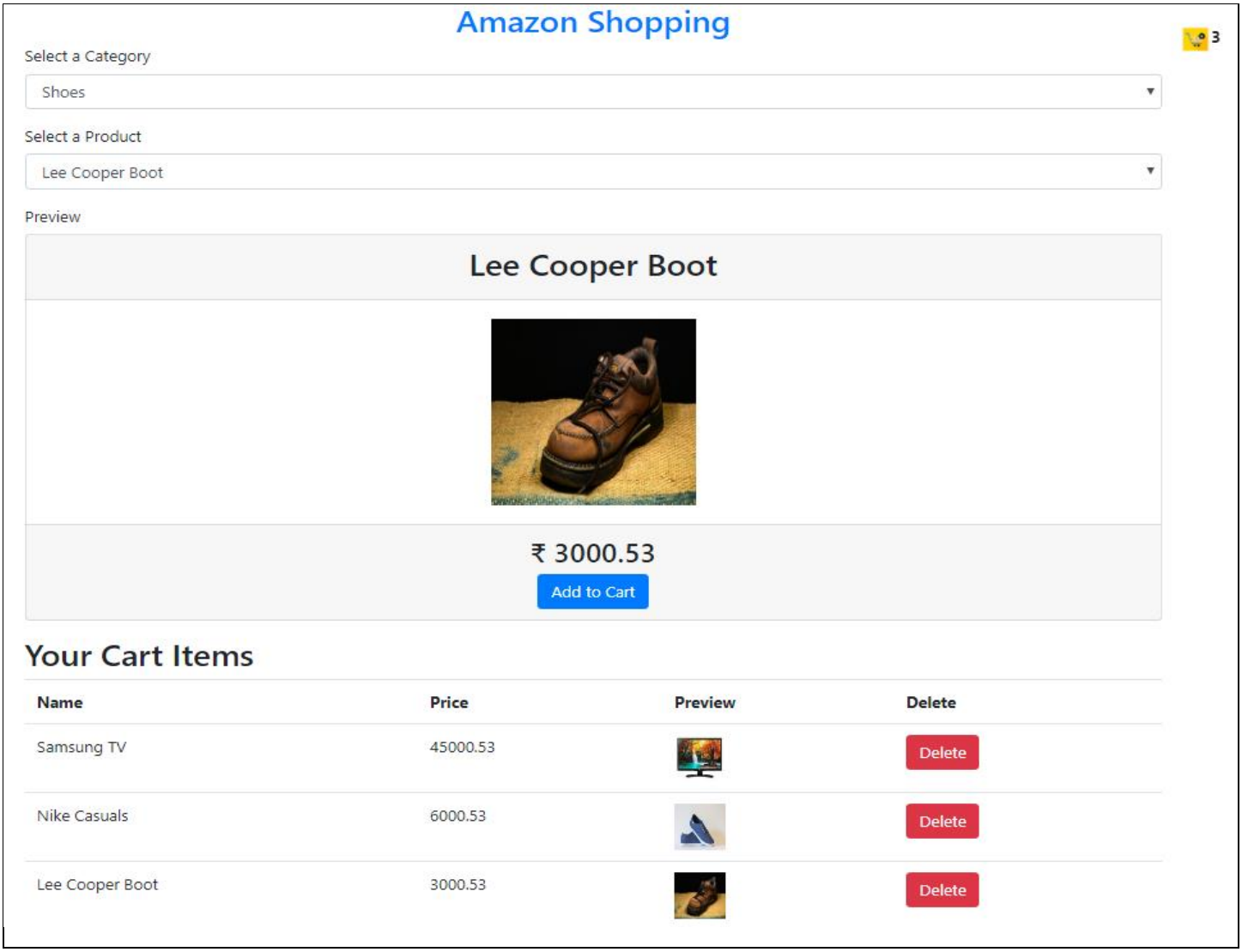
3. Goto “onlineshop.component.html”

```
<div class="container">
    <div (click)="ShowCart()" class="btn" style="position: fixed; right: 10px; to
p:10px">
        <img src="assets/cart.jpg" width="30" height="30">
    <span><b>{{cartItemsCount}}</b></span>
    </div>
    <h2 class="text-center text-primary">Amazon Shopping</h2>
```

```
    <div class="form-group">
      <label>Select a Category</label>
      <div>
      <select (change)="CategoryChanged()" [(ngModel)]="selectedCategory"  class=
"form-control">
              <option *ngFor="let category of categories">
                 {{category}}
              </option>
          </select>
      </div>
    </div>
    <div class="form-group">
       <label>Select a Product</label>
       <div>
       <select (change)="ProductChanged()" [(ngModel)]="selectedProduct" class=
"form-control">
               <option *ngFor="let product of selectedProducts">
                   {{product}}
               </option>
           </select>
       </div>
    </div>
    <div class="form-group">
      <label>Preview</label>
      <div class="card text-center">
       <div class="card-header">
        <h2>{{prodName}}</h2>
       </div>
       <div class="card-body">
         <img [src]="prodPhoto" width="200" height="200" >
       </div>
       <div class="card-footer">
       <h3> &#8377; {{prodPrice}}</h3>
      <button (click)="AddToCart()" class="btn btn-primary"> Add to Cart
       </button>
       </div>
      </div>
    </div>
    <div *ngIf="isCartVisible" class="form-group">
        <h2>Your Cart Items</h2>
        <table class="table table-hover">
           <thead>
               <tr>
                   <th>Name</th>
                   <th>Price</th>
                   <th>Preview</th>
                   <th>Delete</th>
               </tr>
           </thead>
```

```
            <tbody>
                <tr *ngFor="let item of cartItems; let i=index">
                <td>{{item.Name}}</td>
                <td>{{item.Price}}</td>
                <td>
                    <img [src]="item.Photo" width="50" height="50">
                </td>
                <td>
               <button (click)="DeleteCartItem(i)" class="btn btn-danger"> Delete
               </button>
                </td>
                </tr>
            </tbody>
        </table>
    </div>
</div>
```

4. Output



2. Key Event Binding:-

- Angular Support Binding of keyevents to any html element. So that it can defined functionality when user is Keying-in-type character.
- The event are

Event	Description
keyup	Action when key is release
keydown	Action on hold down of key
keypress	Action when new key used after the current

- Event property

Event property	Description
keyCode	It return the actual keyCode, which is ASCII
charCode	It return the character code as per UTF standard
Which	It is similar to keyCode but supported on various keyboed layout
shiftKey ctrlKey altKey	Return true when used.

- Example 18 : - keydemo

Ex.18:- keydemo

1. Add a new Component
   - ng g c keydemo - -spec=false
2. Goto keydemo.component.ts"

```
export class KeydemoComponent  {
  public userList= [
    {userName:'john'},
    {userName:'john_12'},
    {userName:'david'}
  ];
  public userName;
  public msg;
```

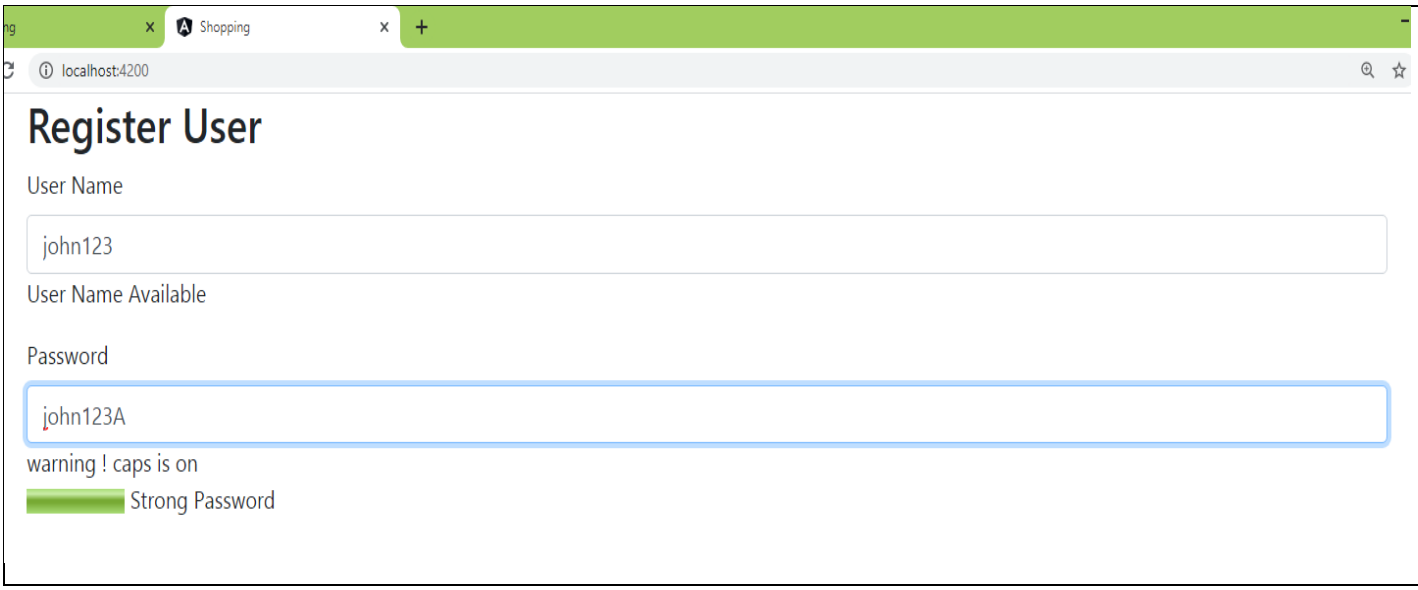
```
public warn;
public pwd;
public regExp = /(?=.*[A-Z])\w{4,15}/;
public pwdStrength='';
public min;
public max;
public low;
public high;
public val;

public strengthMeter(min,max,val,low,high){
  this.min=min;
  this.max=max;
  this.low=low;
  this.high=high;
  this.val=val;
}
public VerifyUser() {
  for(var i=0; i<this.userList.length; i++){
    if (this.userList[i].userName == this.userName){
      this.msg = 'user Name taken-try another';
      break;
    } else {
      this.msg = 'User Name Available';
    }
  }
}
public VerifyCaps(event){
  if(event.keyCode>=65 && event.keyCode<=90){
    this.warn='warning ! caps is on';
  }
  else{
    this.warn='';
  }
}
public VerifyPassword(){
  if (this.pwd.match(this.regExp)){
    this.pwdStrength='Strong Password';
    this.strengthMeter(1,100,100,0,0);
  }
  else{
    if(this.pwd.length<4) {
      this.pwdStrength='poor passsword';
      this.strengthMeter(1,100,100,60,80);
    }
    else{
      this.pwdStrength='weak password';
      this.strengthMeter(1,200,200,40,80);
    }
```

```
    }
  }
}
```

3. Goto “key.component.html”

```
<div class="container">
    <h2>Register User</h2>
    <div class="form-group">
        <label>User Name</label>
        <div>
          <input (keyup)="VerifyUser()" [(ngModel)]="userName" type="text" class=
"form-control">
        <span>{{msg}}</span>
        </div>
    </div>
    <div class="form-group">
        <label>Password</label>
        <div>
                <input (keyup)="VerifyPassword()" (keydown)="VerifyCaps($event)"
[(ngModel)]="pwd"  type="text" class="form-control">
         <span>{{warn}}</span>
         <div>
           <meter min="{{min}}" max="{{max}}" value="{{val}}" low="{{low}}" high=
"{{high}}" > </meter>
            {{pwdStrength}}
            </div>
       </div>
    </div>
</div>
```

4. Output



3. Mouse event binding

- Angular provides a set of event that are used to handled various function. On mouse interaction it include the following

Event	Description
mouseover	Specifies action to perform when pointer is over the element
mouseout/mouseleave	Specified action when mouse pointer leave element.
mousedown	Action when mouse button is down over any element
mouseup	Action when mouse button is relased over any element
mousemove	Action when mouse pointer is move over any element.

-

Event Property	Description
cliientX	Get x-axis position
clientY	Get y-axis position

- Example 19 : - mousedemo (mousemove)
- Example :- mousedemo2(mouseover, mouseout, mousedown, mouseup)

Ex.19:-mousedemo

1. Add a new Component
    a. ng g c mousedemo - -spec=false
2. Goto “mousedemo.component.ts”

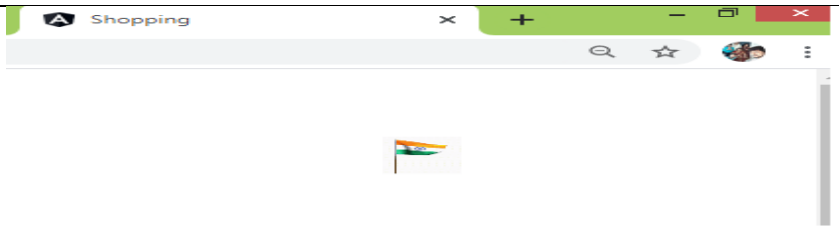
```
export class MousedemoComponent  {
  public styleObj;
  public Animate(event) {
    this.styleObj= {
      'position':'fixed',
      'left':event.clientX + 'px',
```

```
        'top':event.clientY + 'px',
      }
      return this.styleObj;
    }
  }
```

3. Goto “mousedemo.component.html”

```
<body (mousemove)="Animate($event)">
    <div style="height:1000px;"></div>
    <img [ngStyle]="styleObj" src="assets/flag.gif" width="50" height="50">
</body>
```

4. Output



Ex.20:- mousedemo2

1. Add a new Component
    a. ng g c mousedemo2 - -spec=false
2. Goto “mousedemo2.component.ts”

```
export class Mousedemo2Component {
  /**Ex-1 */
  public ad='assets/pepsi1.jpg';
  /**Ex-2 */
  public msg='';
  public styleobj= {
    'width':'100px',
    'height':'100px',
  }
  /**Ex-1 */
  public showAd1(){
    this.ad='assets/pepsi.jpg';
  }
  public showAd2(){
    this.ad='assets/pepsi2.jpg';
  }
  public StartAnimation(e) {
```

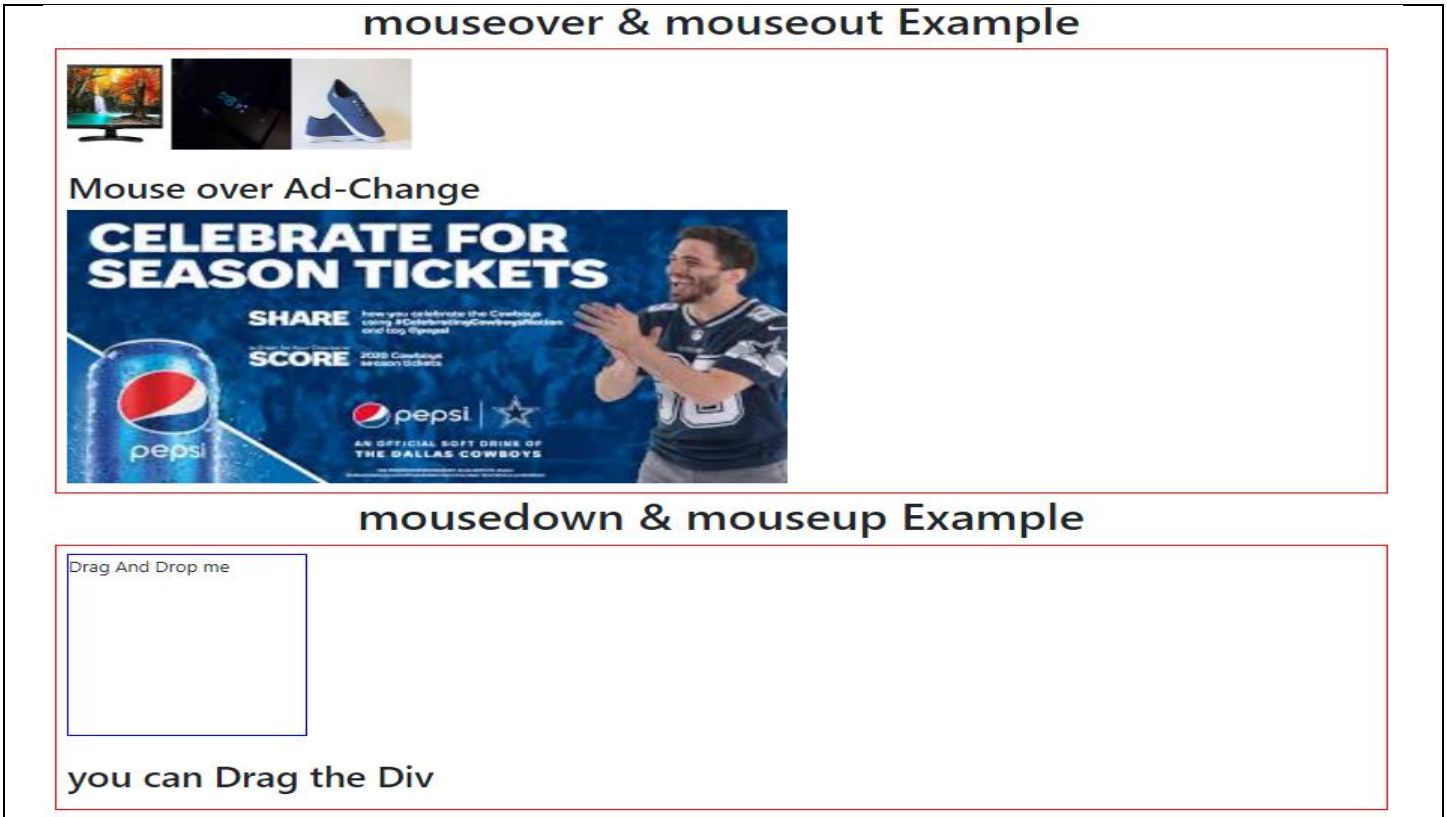
```
    e.target.start();
  }
  public StopAnimation(e) {
    e.target.stop();
  }
  /**Ex-2 */
  public Drag() {
    this.msg='you can Drag the Div';
    this.styleobj = {
      'width':'200px',
      'height':'200px',
    }
  }
  public Drop() {
    this.msg='you can Drop the Div';
    this.styleobj = {
      'width':'100px',
      'height':'100px',
    }
  }
}
```

3. Goto “mousedemo2.component.html”

```
<div class="container">
    <h1 style="text-align: center;">mouseover & mouseout Example</h1>
    <div class="container" style="border: 1px solid red; padding: 10px;">
     <div class="form-group">
          <marquee (mouseover)="StopAnimation($event)" (mouseout)="StartAnimation
($event)" scrollamount="10">
                <div>
                    <img src="assets/tv1.jpg" width="100" height="100">
                    <img src="assets/mobile1.jpeg" width="100" height="100">
                    <img src="assets/shoe1.jpg" width="100" height="100">
                </div>
            </marquee>
        </div>
        <h2>Mouse over Ad-Change</h2>
        <img (mouseover)="showAd1()" (mouseout)="showAd2()" [src]="ad" width="600
" height="300" >
    </div>
    <h1 style="text-align: center;">mousedown & mouseup Example</h1>
    <div class="container" style="border: 1px solid red; padding: 10px;">
        <div [ngStyle]="styleobj" (mousedown)="Drag()" (mouseup)="Drop()" class="
inner-div" style="border: 1px solid blue;">
            <p>Drag And Drop me</p>
        </div>
        <br>
```

```
            <h2>{{msg}}</h2>
    </div>
```

4. Output



4. Miscellaneous event in Angular

1. blur 2. focus	This are the event used to defined the action that are perform when control or element get focus and losses the focus.
3. dblclick	It specified action to perform when your double click on any element
4. cut 5. copy 6. paste	Specified action to perform on cut, copy or paste

- Example 21 : - miscellaneous event

Ex.21:- misceleaneous

1. Add a new Component
    a. ng g c misceleaneous - -spec=false
2. Goto “misceleanoues.component.ts”

```
export class MisceleaneousComponent {
  public status='';
  public uname;
  public msg;
  public styleObj={
    'border-style':'solid',
    'border-color':'black',
    'border-width':'1px',
  }
  public ChangeCase() {
    if(this.uname=='') {
      this.msg='name is required';
      this.styleObj={
        'border-style':'solid',
        'border-color':'red',
        'border-width':'2px',
      }
    }
    else {
      this.uname=this.uname.toUpperCase();
      this.styleObj={
        'border-style':'solid',
        'border-color':'green',
        'border-width':'2px',
      }
      this.msg
    }
  }
  public ShowMsg(){
    this.msg="Name in block letter";
  }
  public ViewLarge() {
    window.open('assets/tv1.jpg','tv','width=600 height=500')
  }
  public OnCut() {
    this.status='REmoved-copied to clipbord'
  }
  public OnCopy() {
    this.status='copied to clipbord'
  }
```

```
  public OnPaste() {
    this.status='Inserted from clipbord'
  }
}
```

3. Goto “misceleanoues.component.html”

```
<div class=container>
    <table class="table table-hover">
        <thead>
            <tr>
                <th>Blur and focus</th>
                <th>cut, copy and paste</th>
                <th>double click (dblclick)</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td><label>User Name</label></td>
                <td><label>Work here</label></td>
                <td><label> double Click on pic</label></td>
            </tr>
            <tr>
                <td><input [ngStyle]="styleObj" [(ngModel)]="uname" (blur)="Chang
eCase()" (focus)="ShowMsg()" type="text" placeholder="Name in block letter"></td>
                <td><input (cut)="OnCut()" (copy)="OnCopy()" (paste)="OnPaste()"
type="text"></td>
                <td><img (dblclick)="ViewLarge()" src="assets/tv1.jpg" width="40"
 height="40" ></td>
            </tr>
            <tr>
                <td>msg:- <span>{{msg}}</span></td>
                <td>msg: <span>{{status}}</span></td>
                <td>msg:- on dbl click new window come</td>
            </tr>
        </tbody>
    </table>
</div>
```

4. Output

5. Submit Event

- Submit is an event used by <form> element is order to identify "Form Submit" and handle a functionality on submit.
- The submit event fireup on submit button click.
- A submit button can defined by using

  a) `<input type="submit">`

  b) `<button></button>`
- Submit button is a generic button that by default have the ability to submit the form data.
- Form submit data as QueryString on get request and it submit as form body on post request.
- Example 22 : - submit button

Ex.22:- submit event

1. Add a new Component
   - ng g c submit - -spec=false
2. Goto "submit.component.ts"

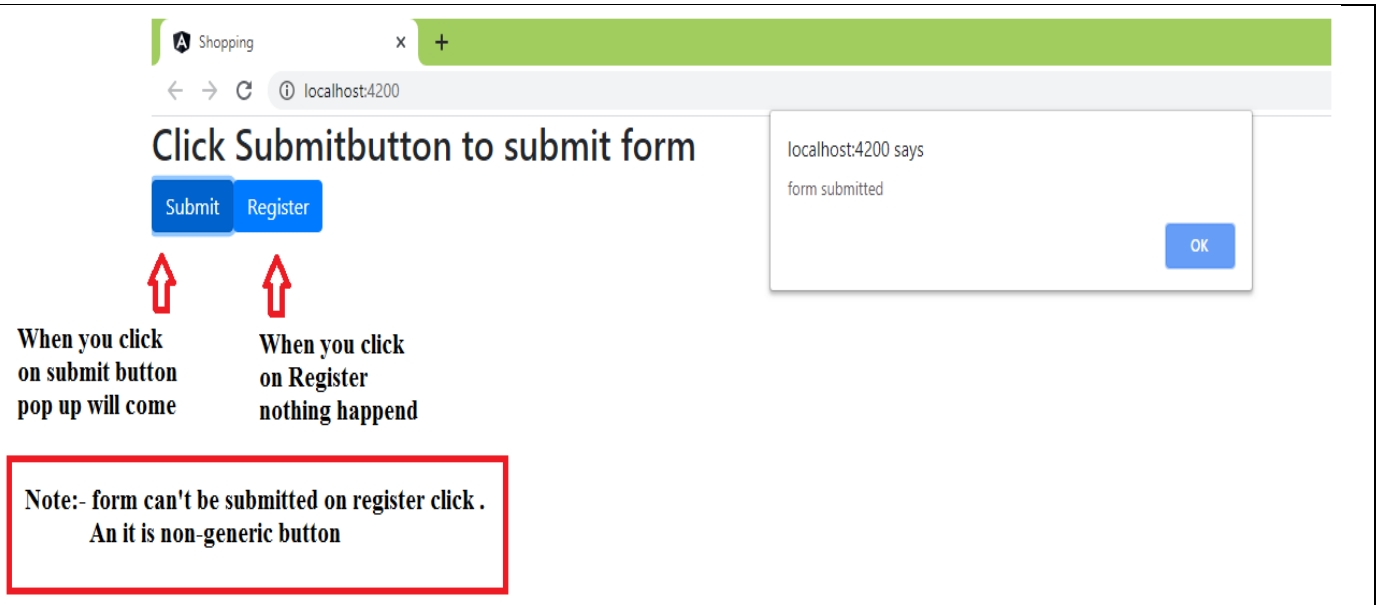
```
import { Component, OnInit } from '@angular/core';
```

```
@Component({
  selector: 'app-submit',
  templateUrl: './submit.component.html',
  styleUrls: ['./submit.component.css']
})
export class SubmitComponent  {
  public SubmitClick() {
    alert('form submitted')
  }
}
```

3. Goto “submit.component.html”

```
<form method="GET" (submit)="SubmitClick()">
    <h2>Click Submitbutton to submit form</h2>
    <button class="btn btn-primary">Submit</button>
    <input type="button" value="Register" class="btn btn-primary">
</form>
```

4. Output



FAQ:-

1. **Can a form have multiple submit buttons ?**

   Ans: - Yes

2. **How submit identifies specific button click?**

   Ans: - by accessing the button value on submit.

3. **Can we submit form on any another event used by HTML element.** Ans: - Yes

Component Hierarchy

Component hierarchy

- Component are the building block for angular application.
- A component can access another component within the contact by using parent→Child hierarchy.
- Any component can be access within the contacts by using a selector.
- Syntax

```
        index.html
<app-parent></app-parent>
        parent.html
<app-child></app-child>
```

- The property of any child component are not directly accessible to the parent component.
- You have to flag the properties of child component by using the directive "@Input( )".
- The properties that are flaged with Input ( ) are accessible to parent component by using property binding technique.
- Syntax

```
<app-child [property]=""></app-child>
```

- Example 23 Parentdemo and childdemo
- Example 24. Nested component with input and output attribute, and custom events.
  - A) by radio button
  - B) by check box
  - C) by selector <option>

Ex.23: - parntdemo & childdemo

1. Add a new Component
   - ng g c parentdemo - -spec=false
   - ng g c childdemo –spec=false
2. Goto "childdemo.component.ts"

```
export class ChilddemoComponent  {
  @Input() public msg = "Hello ! Form child";
}
```

3. Goto “childdemo.component.html”

```
<div>
    <h4>child component</h4>
    {{msg}}
</div>
```

4. Goto “parentdemo.component.ts”

```
export class ParentdemoComponent  {
  public parentmsg ="Hi ! form parent";
}
```

5. Goto “parentdemo.component.html”

```
<div class="container">
    <h2>parent component</h2>
    <app-childdemo [msg]="parentmsg"></app-childdemo>
</div>
```

6. Output

Ex.24: - productdata & productfilter

1. Add a new Component
   - ng g c productsdata - -spec=false
   - ng g c productsfilter –spec=false
2. Goto “productdata.component.ts”

```
export class ProductsdataComponent  {
  products = [
    {Name:'Samsung TV',Price:45000, Category:'Electronics'},
    {Name:'Mobile',Price:15000, Category:'Electronics'},
```

```
    {Name:'Nike casual',Price:5300, Category:'Shoes'},
    {Name:'Lee cooper',Price:8500, Category:'Shoes'},
  ];
  public AllCount = this.products.length;
  public ElectronicsCount = this.products.filter(x=>x.Category == 'Electronics').
length;
  public ShoesCount = this.products.filter(x=>x.Category == 'Shoes').length;

  public selectedCategoryValue = 'All';
  public CategoryChanged(selectedCategoryName){
    this.selectedCategoryValue  = selectedCategoryName;
    console.log(this.selectedCategoryValue)
  }
}
```

3. Goto “productsdata.component.html”

```
<div class="container">
    <h2>Products List</h2>
    <div class="form-group">
        <app-
productsfilter (ComponentChangedEvent) = "CategoryChanged($event)" [AllCount]="Al
lCount" [ElectronicsCount] = "ElectronicsCount" [ShoesCount] = "ShoesCount">
        </app-productsfilter>
    </div>
    <div class="form-group">
        <table class="table table-hover">
            <thead>
                <tr>
                    <th>Name</th>
                    <th>Price</th>
                    <th>Category</th>
                </tr>
            </thead>
            <tbody>
                <ng-container *ngFor = "let product of products">
                    <tr *ngIf = "selectedCategoryValue=='All'|| selectedCategoryV
alue == product.Category">
                        <td>{{product.Name}}</td>
                        <td>{{product.Price}}</td>
                        <td>{{product.Category}}</td>
                    </tr>
                </ng-container>
            </tbody>
            </table>
    </div>
</div>
```

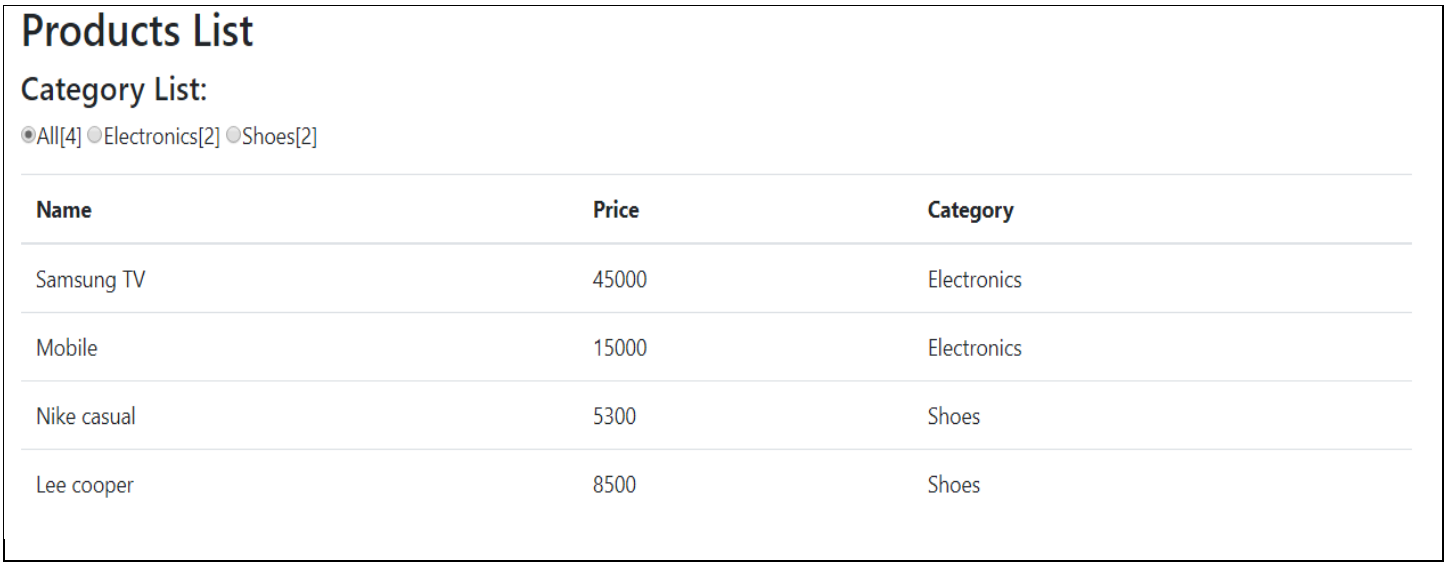
4. Goto “productsfilter.component.ts”

```
export class ProductsfilterComponent {
  @Input() public AllCount = 0;
  @Input() public ElectronicsCount = 0;
  @Input() public ShoesCount = 0;

  public selectedCategoryValue = 'All';

  @Output() public ComponentChangedEvent:EventEmitter<string> = new EventEmitter<
string>();

  public RadioChanged(){
    this.ComponentChangedEvent.emit(this.selectedCategoryValue);
  }
}
```

5. Goto “productsfilter.component.html”

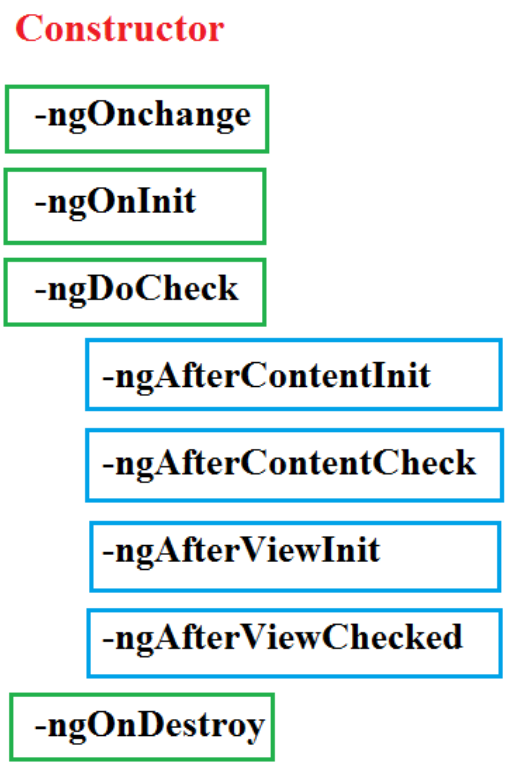
```
<div>
    <h4>Category List:</h4>
    <input (change) = "RadioChanged()" [(ngModel)] = "selectedCategoryValue"  typ
e="radio" name="category" value="All">All[{{AllCount}}]
    <input (change) = "RadioChanged()" [(ngModel)] = "selectedCategoryValue"  typ
e="radio" name="category" value="Electronics">Electronics[{{ElectronicsCount}}]
    <input (change) = "RadioChanged()" [(ngModel)] = "selectedCategoryValue"  typ
e="radio" name="category" value="Shoes">Shoes[{{ShoesCount}}]
</div>
```

6. Output



Component life cycle event

- A component is the core feature of angular application.
- A component have specific life cycle, which is manager by angular.
- Angular creates the component on request.
- It render the component to the client
- It create and render its client component.
- It verifies the changes of properties and value in a component.
- It destroy the component related memory before removing from DOM.
- The various phases of component cycle as show below.



Life cycle event	Description
ngOnInit	Its specified the action to perform when component is initialized.
ngOnChange	Its specified the action to perform when any changed accured in the component.

ngDoCheck	Its specifies the event, which is used to identify the changes explicitly. i.e. the action to perform explicitly when component changed.
ngOnDestroy	Its specifies the action to perform when component is removed form DOM hierarchy.

- Ex.25 :- tracking the changes of parent component in child component

Ex.25: - tracking the changes of parent component in child component.

1. Add a new Component
   - ng g c parent - -spec=false
   - ng g c child –spec=false
2. Goto “parent.component.ts”

```
export class ParentComponent  {
  public uname;
}
```

3. Goto “parent.component.html”

```
<div class="container">
    <h2>parent</h2>
    <input type="text" [(ngModel)]="uname">
    <app-child [uname]="uname"></app-child>
</div>
```

4. Goto “child.component.ts”

```
import { Component, OnChanges, Input, SimpleChanges } from '@angular/core';

export class ChildComponent implements OnChanges {
  @Input() public uname;
  public currentValue;
  public previousValue;
  ngOnChanges(changes:SimpleChanges){
    for(var property in changes){
      let change = changes[property];
      this.currentValue=change.currentValue;
      this.previousValue=change.previousValue;
    }
  }
}
```

5. Goto “child.component.html”

```
<div>
    <h2>Hello ! {{uname}}</h2>
    <dl>
        <dt>Current value</dt>
        <dd>{{currentValue}}</dd>
        <dt>previous value</dt>
        <dd>{{previousValue}}</dd>
    </dl>
</div>
```

6. Output

- **Summary**

Angular Pipes.

- Pipes is a function in angular used to format the data dynamically.
- Technically pipe are derived from pipe transform defined in “angular/core” library.
- Pipe uses a “transform ( )” which takes input of any value dynamically, transform the value into specified format and return to view .
- The metadata of a pipe is defined by using the directive @pipe ( )
- The predefined pipes available in angular library are
  - Uppercase
  - Lowercase
  - Number
  - Currency
  - Percent
  - Data
  - Slice …..etc

1. Uppercase and lowercase: -

- These are the angular pipe used to convert the string into uppercase and lowercase letters
- Syntax: -

```
public msg="Welcome";
<div>{{msg | uppercase}}</div>
```

- Output: - WELCOME
- Angular pipe are defined with parameter the parameterize pipes will accept parameter by using the separator colon (:) you can concatenate multiple pipes by using pipes symbol ‘|’
- Syntax: -

```
{{ data | pipeName : params | pipeName……  }}
```

2. Number / decimal:-

- These are the used to display numeric value with thousand separator and decimal places.
- It requires the parameter with specified the number of factions.
- Syntax: -

```
{{ expression | number : ‘decimalplaces.min-max’ }}
```

3. Currency: -

- It is an angular pipe similar to number but can display a numeric value. With decimal places and currency symbol.
- Syntax: -

```
{{expression | currency : ‘symbol’ : ‘decimalplaces.min-max’}}
```

- Example:-

```
public price = 42000.50;
<dt>Price</dt>
<dd>{{price}}</dd>                                  // 42000.5
<dd>{{price | currency}}</dd>                       // $42,000.50
<dd>{{price | currency : 'INR'}}</dd>               //₹42,000.50
<dd>{{price | currency : '&#8377;'}}</dd>           //₹42,000.50
```

4. Date pipe : -

- It is an angular filter used to display the data value in a specific date format which includes both short date and long date.
- Syntax: -

```
{{ expression | date : ‘format’ }}
```

- Predefined format string
  - Short
  - Long
  - Medium
- Predefined format style: -

M	-	Month number single digit
MM	-	Month number 2 digit
MMM	-	Month number short format
MMMM	-	Month number long format
d	-	Day number single digit
dd	-	Day number 2 digit
yy	-	Year number 2 digit

yyyy	-	Year number 4 didgit

- Example:-

```
public mfd = new Date("01/02/2020")
<dt>Date</dt>
<dd>{{mfd | date}}</dd>                  //Jan 2, 2020
<dd>{{mfd | date : 'long'}}</dd>        //January 2, 2020 at 12:00:00 AM GMT+5
<dd>{{mfd | date : 'd MMMM yyyy'}}</dd> //2 January 2020
```

5. Percent pipe: -

- It is an angular pipe used to display numeric value in percent format by using percent symbol. It can also used the parameter that specified decimal places and fraction.
- Syntax: -

```
{{ expression | date : ‘format’ }}
```

- Example: -

```
public sale = 0.265;
<dt>Sale</dt>
<dd>{{sale | percent}}</dd>  //27%
```

6. Slice: -

- It is an angular pipe used to slice any string based on specified index value.
- Syntax: -

```
{{ string | slice : start Index : end Index }}
```

- Example: -

```
public msg= "Welcome to Angular"
<dt>Slice</dt>
<dd>{{msg | slice:0:7}}</dd>
```

- You can create custom pipe by implementing pipe transform and configuration your own functionality by using “transform ( )”
- Example: 1- Angular pipe
- Example: 2- Sentence case
- Example: 3- Title case

Ex.1: - angularpipe

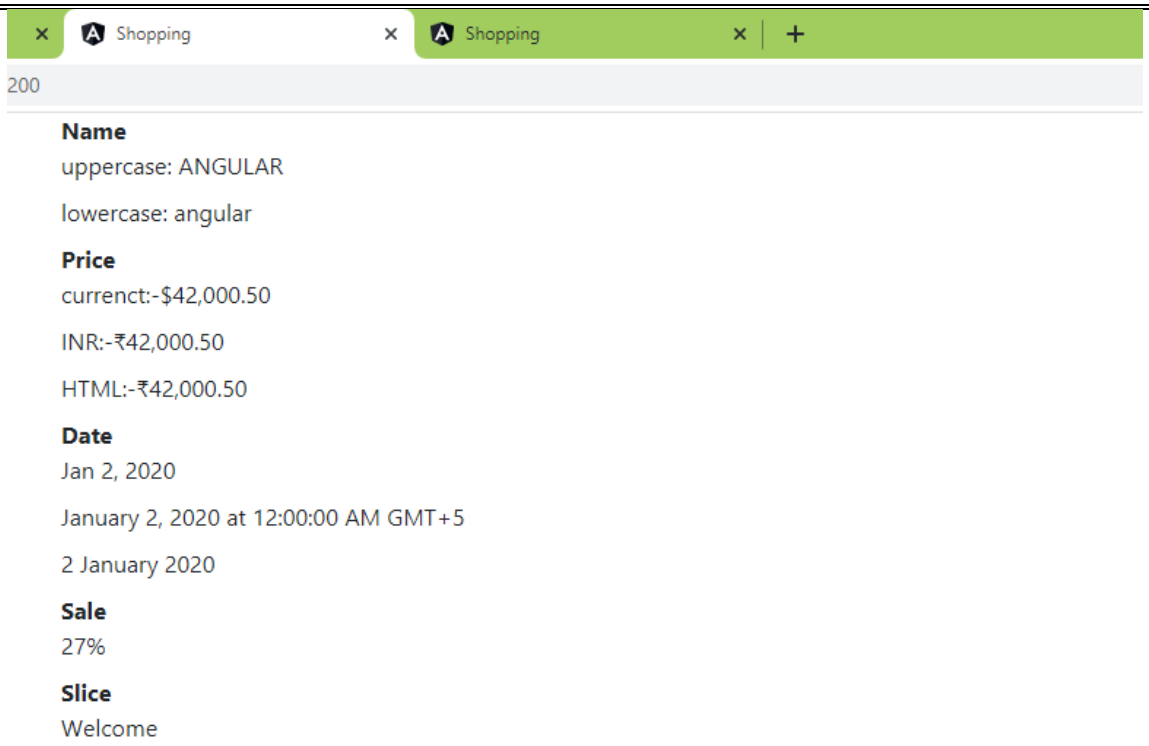
1. Add a new Component
   - ng g c angularpipe - -spec=false
2. Goto “angularpipe.component.ts”

```
export class AngularpipeComponent  {
  public msg=' Angular';
  public price = 42000.50;
  public mfd = new Date("01/02/2020")
  public sale = 0.265;
  public slice= 'Welcome to Angular';
}
```

3. Goto “angularpipe.component.html”

```
<div class="container">
    <dl>
        <dt>Name</dt>
        <dd>uppercase: {{msg | uppercase}}</dd>
        <dd>lowercase:{{msg | lowercase}}</dd>
        <dt>Price</dt>
        <dd>currenct:-{{price | currency}}</dd>
        <dd>INR:-{{price | currency : 'INR'}}</dd>
        <dd>HTML:-{{price | currency : '&#8377;'}}</dd>
        <dt>Date</dt>
        <dd>{{mfd | date}}</dd>
        <dd>{{mfd | date : 'long'}}</dd>
        <dd>{{mfd | date : 'd MMMM yyyy'}}</dd>
        <dt>Sale</dt>
        <dd>{{sale | percent}}</dd>
        <dt>Slice</dt>
        <dd>{{slice | slice:0:7}}</dd>
    </dl>
</div>
```

4. Output



EX.2: - custom pipe (sentencecase)

1. Add a new file into "app" folder by name

   "sentencecase.pipe.ts"

2. Sentencecase.pipe.ts

```
import { Pipe, PipeTransform } from '@angular/core';
@Pipe({
    name:'sentencecase'
})
export class SentenceCasePipe implements PipeTransform{
    transform(str){
        let firstChar=str.charAt(0);
        let restChars=str.substring(1);
        let sentence=firstChar.toUpperCase()+ restChars.toLowerCase();
        return sentence;
    }
}
```

3. Register pipe in "app.module.ts"

```
  declarations: [
    SentenceCasePipe,
  ],
```

4. Implement in your page

```
public msg=' welcome to angular';
{{msg | sentancecase}}
```

5. Output

```
Welcome to angular
```

Ex.3: - title case

1. Add a new file into “app” by name
   - titlecase.pipe.ts
2. Goto “titlecase.pipe.ts”

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
  name: 'titilecase'
})
export class TitilecasePipe implements PipeTransform {

  transform(str) {
    let firstchar = str[0].toUpperCase();
    let restchar = str.slice(1).toLowerCase();
    let capitalstring = (str) => firstchar + restchar;
    return capitalstring
  }
}
```

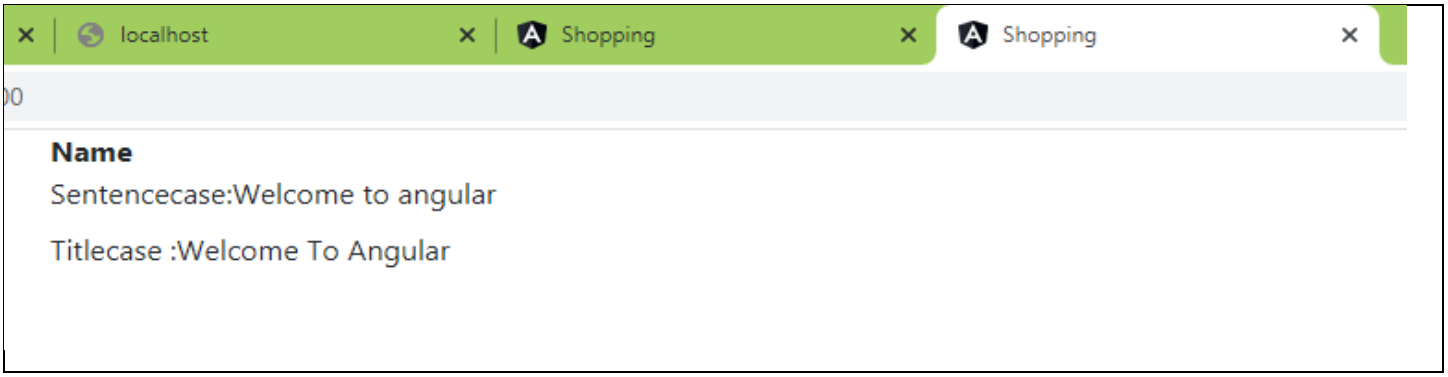
3. Register pipe in “app.module.ts”

```
declarations: [
    TitleCasePipe
]
```

4. Implement in your page.

```
<div class="container">
    <dl>
        <dt>Name</dt>
        <dd>Sentencecase:{{msg | sentencecase}}</dd>
        <dd>Titlecase :{{msg | titlecase}}</dd>
    </dl>
</div>
```

5. Output



Summary

Angular Services.

- A service is predefined business logic which can be reused in the application by injecting into component.
- Technically service is a collection of factories and the factory is collection of related type of functions.
- Factory uses a single call mechanism where a new object is required every time in order to access any functionality.
- Services uses a single tern mechanism where an object is created only ones and the same object is used across multiple request.
- You can access the function any number of time by using single object.
- Services in angular is a class defined with set of function that return specified functionality.
- A services can be injected into any component by configuring meatadata using @injectable ( ) directive.
- Service are register into angular application at “provider” in app.module.ts.
- Example 1.
- Example : 2- Editing services for products Data

Ex.1: - captcha service

1. Add a new file into project by name
   - Captcha.service.ts
2. Goto “captcha.service.ts”

```
import { Injectable } from '@angular/core';

@Injectable ()
export class CaptchaService{
    public GenerateCode() {
        let a = Math.random() * 10;
        let b = Math.random() * 10;
        let c = Math.random() * 10;
        let d = Math.random() * 10;
```

```
        let e = Math.random() * 10;
        let code = Math.round(a) +' '+ Math.round(b) +' '+ Math.round(c) +'
 '+ Math.round(d)+' '+ Math.round(e);
        return code;
    }
}
```

3. Register sevice in “app.module.ts”

```
import { CaptchaService } from './captcha.servive';
providers: [CaptchaService]
```

4. Add a new component

   - ng g c login –spec=false

5. Goto ”login.component.ts”

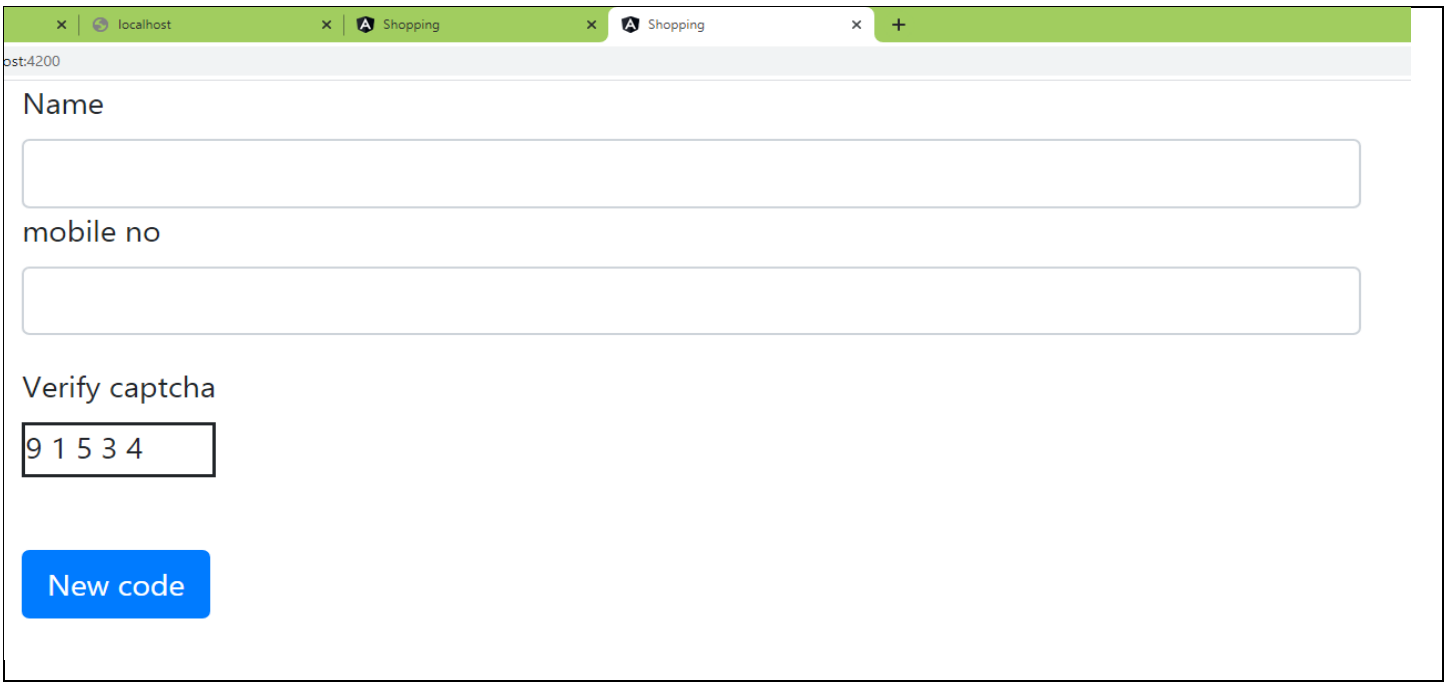
```
import { Component, OnInit } from '@angular/core';
import { CaptchaService } from '../captcha.servive';

export class LoginComponent implements OnInit {
  public code;
  constructor (private captcha:CaptchaService){

  }
  ngOnInit(){
    this.code=this.captcha.GenerateCode();
  }
}
```

6. Goto ”login.component.html”

```
<div class="container">
    <div class="form-group">
        <label>Name</label>
        <input type="text" class="form-control">
        <label> mobile no</label>
        <input type="text" class="form-control">
    </div>
    <div class="form-group">
        <label>Verify captcha</label>
        <div style="width:100px; height:30px; border-style:solid;">
        {{code}}</div>
    </div>
    <br>
    <div>
        <button (click)="ngOnInit()" class="btn btn-primary">New code
```

```
            </button>
        </div>
    </div>
```

7. Output



Ex.2: - apidata- Editing service for products data (_Same ex.no.23)

1. Add a new service
   - ng g service apidata - -spec=false
2. Goto "apidata.service.ts"

```
import { Injectable } from '@angular/core';

@Injectable({
  providedIn: 'root'
})
export class ApidataService {
  public GetProducts(){
    return[
      {Name:'Samsung TV',Price:45000, Category:'Electronics'},
      {Name:'Mobile',Price:15000, Category:'Electronics'},
      {Name:'Nike casual',Price:5300, Category:'Shoes'},
      {Name:'Lee cooper',Price:8500, Category:'Shoes'},
    ];
  }
}
```

3. Register service in “app.module.ts”

```
providers: [ApidataService]
```

4. Inject in your component

- “productsdata.component.ts”

```
import { ApidataService } from '../apidata.service';

export class ProductsdataComponent implements OnInit {
  constructor(private data:ApidataService){
  }
  products = [];
  ngOnInit(){
    this.products = this.data.GetProducts();
  }
```

5. Output – same as example no.23 in angular component

Summary

Angular Forms

- Form is a container that encapsulate a set of element and provide an UI form where can interact with our application.
- Technically form compresses of element like button, text box, check box, radios, dropdown list etc….
- A form allows the user to query and submit date to service.
- Angular can give a dynamic behavior forms. So that the can handle client side interaction and validations.
- Angular form are classified into 2 types
    1. Template Driven forms
    2. Reactive forms or model driven forms.

1. Template driven forms

- A template driven forms is a dynamic forms that can handle client side form is defined at template level. i.e. in html.
- Most of the interactions are dynamically handled at UI level.
- These forms are heavy on page and will take more time rendering (generating o/p).
- The form and their elements in template are configure by using the directive.
    - ngForm
    - ngModel
- ngForm is defined in “forms model” and used to handle <form> element dynamically.
- ngModel is also defined in “forms Model” and used to handled input element dynamically. [like text, password, select etc…]
- Syntax :-

```
<form #referenceName = "ngForm">
<input type="text" #txtName = "ngModel" ngModel name="txtName">
</form>
```

- Example:- Templateform 1

Ex.1: - templateform – basic

1. Goto “app.module.ts”

```
import { BrowserModule } from '@angular/platform-browser';
import { FormsModule } from '@angular/forms';

imports: [
BrowserModule,
FormsModule
],
```

2. Add a new Component
   - ng g c templateform - -spec=false
3. Goto “templateform.component.ts”

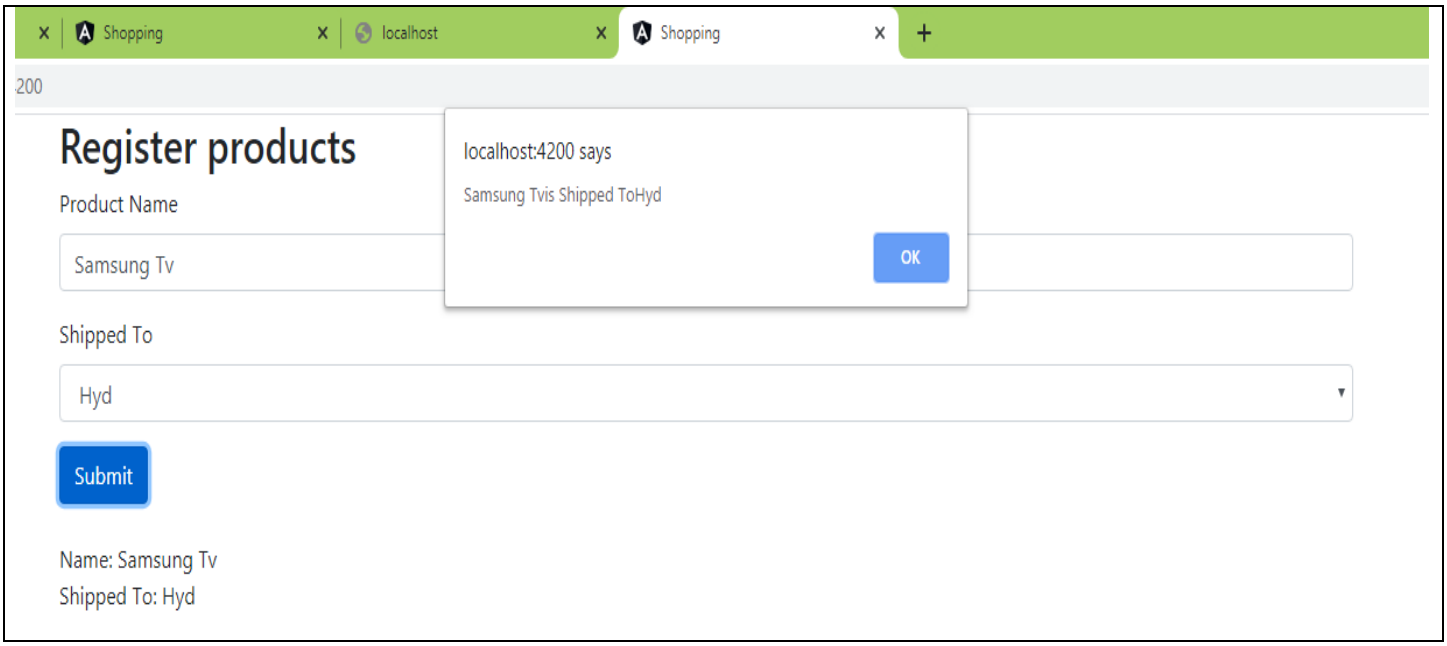
```
export class TemplateformComponent  {
  public SubmitClick(obj){
    alert(obj.txtName + 'is Shipped To' + obj.ShippedTo);
  }
}
```

4. Goto “templateform.component.html”

```
<div class="container">
    <h2>Register products</h2>
    <form #frmRegister="ngForm" (submit)="SubmitClick(frmRegister.value)">
        <div class="form-group">
            <label>Name</label>
            <input type="text" class="form-
control" name="txtName" required #txtName="ngModel" ngModel>
        </div>
        <div class="form-group">
            <label>Shipped To</label>
            <div>
                <select class="form-
control" name="ShippedTo"#shippedTo="ngModel" ngModel>
                    <option>Hyd</option>
                    <option>Chenni</option>
                </select>
            </div>
        </div>
        <button class="btn btn-primary">Submit</button>
    </form>
    <br>
```

```
        Name: {{frmRegister.value.txtName}}
        <br>
        Shipped To: {{frmRegister.value.ShippedTo}}
    </div>
```

5. Output



Angular form Validation

- Validation is the process of verifying user input.
- Validation is required to insured that contradictory and unauthorized data is not get stored into the database.
- Client side validations can be handled by using patterns, regular expressions and functions.
- Angular provides predefined services to handle validation.
- The angular validation services are categories into two types.
    - Form state validation services.
    - Input state validation services.

1. Form state services:-

- The form state validation services can verify all input field in a form simultaneously
- At the same time and return a Boolean value
- The varies form date services are :

Service	Property	Description
ngPristine	pristine	It verifies whether any field in the form have modified its value with user input its return true when no field modified.
ngDirty	dirty	It return Boolean true if any one field has been modify.
ngInvalid	invalid	It return true when at least one field is invalid.
ngValid	valid	It return true only when all field are valid.
ngSubmitted	submitted	It return true only when the form is submitted.

Ex.2. Templeform 2

1. Add a new Component
   - ng g c templateform - -spec=false
2. Goto “templateform.html”

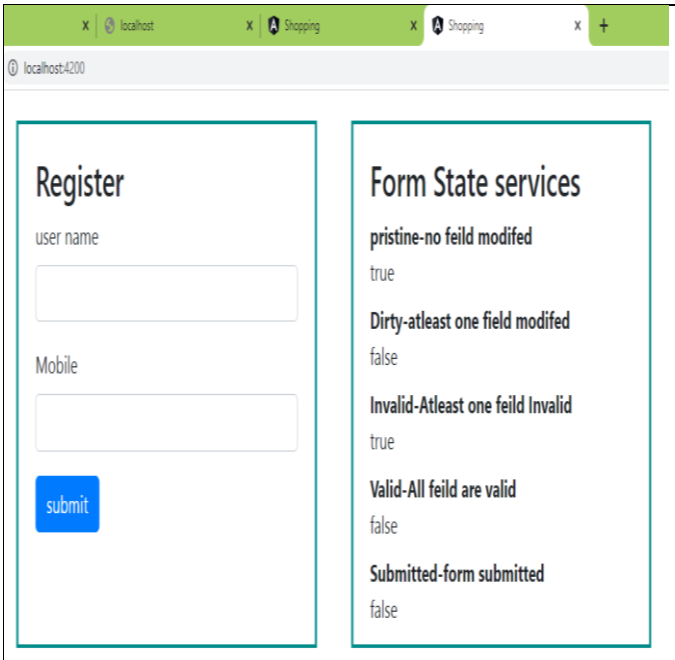
```
<div class="container">
    <div class="boxStyle">
        <h3>Register</h3>
        <form novalidate name="frmRegister" #frmRegister="ngForm">
            <div class="form-group">
                <label>user name</label>
                <div>
                    <input type="text" name="txtName" ngModel #txtName="ngModel"
class="form-control" required>
                </div>
            </div>
            <div class="form-group">
                <label>Mobile</label>
                <div>
                  <input type="text" name="txtMobile" ngModel #txtmobile="ngModel"
 class="form-control" required pattern="\+91[0-9]{10}">
                </div>
            </div>
            <div class="form-group">
<button [disabled]="frmRegister.Invalid" class="btn btn-primary" >submit</button>
            </div>
        </form>
    </div>
```

```
        <div class="boxStyle">
            <h3>Form State services</h3>
            <dl>
                <dt>pristine-no feild modifed</dt>
                <dd>{{frmRegister.pristine}}</dd>
                <dt>Dirty-atleast one field modifed</dt>
                <dd>{{frmRegister.dirty}}</dd>
                <dt>Invalid-Atleast one feild Invalid</dt>
                <dd>{{frmRegister.invalid}}</dd>
                <dt>Valid-All feild are valid</dt>
                <dd>{{frmRegister.valid}}</dd>
                <dt>Submitted-form submitted</dt>
                <dd>{{frmRegister.submitted}}</dd>
            </dl>
        </div>
    </div>
```

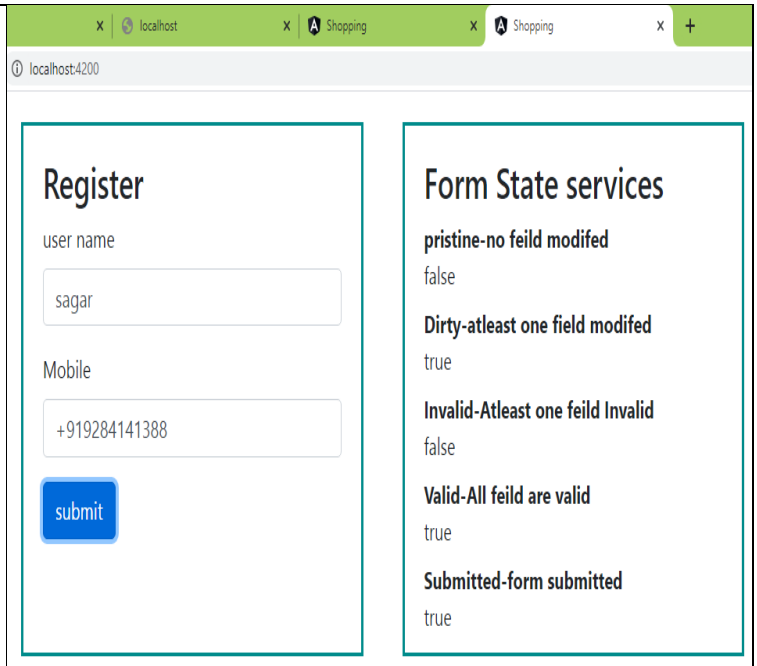
3. Goto " templateform.component.css"

```
.boxStyle{
    width: 350px;
    height: 350px;
    float: left;
    margin: 20px;
    padding: 20px;
    border: 2px solid darkcyan;
}
```

4. Output



Before check value



After check value

2. Input state validation services : -

- Angular provides predefined services that are used to verify the state of every field individually.

Service	Property	Description
ngTouched	touched	It returns true when form element gets focus and the blurred.
ngUnTouched	untouched	It returns true when the input field never focus.
ngPristine	pristine	it return true if field is not modified.
ngDirty	dirty	It return true if field is modify.
ngInvalid	invalid	It return true if field is invalid.
ngValid	valid	It return true if field is valid.
ngSubmitted	submitted	It return true only when the form is submitted.
erroes	errors	It is an input field object that contains collection of error properties. That are used to verify specific error in the field.

- Syntax:-

```
fieldName.invalid

fieldName.error.required
```

- Note:- error object can be invoked only when input field is invalid.
- Example 1 : input validation 1

Dynamically apply CSS effects for validation

- You can defined CSS classes for validation errors.
- You can dynamically apply classes by using “ngClass” directive.
- It required to use an object reference to turn ON or OFF any classes.

- The Boolean value is send by using validation services.
- Syntax:- `<input [ngClass]="{className : txtName.value}">`
- Example 2 : input validation 2

Angular Built-in validation CSS classes.

- Angular provides a set of "predefined" CSS classes for validation.
- The built-in classes can identify the state and apply effects automatically to any specified element.
- The built in classes doesn't have any predefined effects you have to configure the effects manually [explicitly]
- The angular CSS classes are :

Class name	Description
.ng-valid	Applies effect when the input state is valid.
.ng-invalid	Applies effect when input state is invalid
.ng-pristine	Applies effect when the input field not yet modified.
.ng-dirty	Applies effect when input field is modify.
.ng-touched	Applies effect when input field touched.
.ng-untouched	Applies effect when input field untouched.

- Example 3 : input validation 3
- Note:- you can apply predefined CSS angular validation classes for "form state validation"
- Syntax:

```
form.ng-invalid{
background-color: darkcyan;
}
```

Custom validation in angular: -

- HTML provide a limited set of validation properties like required, email, url, minlength etc.
- The input value can be verified by using the property provide by HTML.
- However various other validations need to be defined by custom function and events.
- You can create custom validation by accessing the input value on any specific event and verify with the required value.
- The errors masses can be display by using a custom Boolean property that identifies the validation error.
- Example 4 : custom validation

Custom validation with string function:-

- Example 5 : custom validation (previous example)
- Example 6 : changing validation pattern dynamically for any Element.
-